DEFENSE INFORMATION SYSTEMS AGENCY

ANALYSIS OF WEB

SERVICES STANDARDS

FINAL

13 NOVEMBER 2003

Table of Contents

Executive Summary ..... ES 1
ES.1 Purpose and Scope ..... ES 1
ES.2 Major Findings ..... ES 1
ES.3 Recommendations ..... ES 3
ES.4 Conclusions ..... ES 4
1 Introduction 1
1.1 Conventions 1
1.2 How to Read this Document 1
1.3 Section Overview 2
2 Web Services 4
2.1 Web Services defined 4
2.2 Characteristics of Web Services 4
2.2.1 The Ideal Middleware 5
2.2.2 Web Services as Ideal Middleware 5
2.3 Foundations of Web Services 6
2.3.1 Extensible Markup Language (XML) 6
2.3.2 XML Schema 8
2.3.3 Simple Object Access Protocol (SOAP) 9
2.3.4 Web Services Description Language (WSDL) 11
3 Web Services Architectures 14
3.1 Message Exchange Patterns 14
3.1.1 One-Way 14
3.1.2 Request/Response 14
3.1.3 Solicit Response 15
3.1.4 Notification 15
3.2 Architecture Types 16
3.2.1 Service Oriented Architecture (SOA) 16
3.2.2 Peer-To-Peer (P2P) 17
3.2.3 Enterprise Bus 17
3.2.4 Grid Computing 18
3.3 Specific Architectures 18

3.3.1 W3C Web Services Architecture (WSA) 19
3.3.2 Global XML Web Services Architecture (GXA) 22
3.3.3 Electronic Business XML (ebXML) 24
3.3.4 General Recommendations 27
3.4 References 27
4 Web Services and Messaging 29
4.1 Transfer Protocols 29
4.1.1 HTTP and Other Known Transfer Protocols 29
4.1.2 Message-Oriented Middleware (MOM) 30
4.1.3 Recommendation 30
4.2 Messaging and Attachments 30
4.2.1 Specification and Status 30
4.2.2 Main Concepts 31
4.2.3 Recommendation 31
4.3 Reliable Messaging 31
4.3.1 Specification and Status 32
4.3.2 Main Concepts 32
4.3.3 Recommendation 33
4.3.4 WS-Routing/WS-Referral (GXA) 33
4.3.5 Binary XML and XML Compression 35
5 Web Services and Security 37
5.1 Security Framework - OASIS Web Services Security (WS-Security) 38
5.1.1 Specification and Status 38
5.1.2 Main Concepts 38
5.1.3 Assessment 40
5.1.4 Implementations 41
5.1.5 Recommendation 41
5.2 Authentication/Identity Management 41
5.2.1 OASIS Security Assertion Markup Language (SAML) 41
5.2.2 Liberty Alliance 46
5.2.3 WS-Federation (GXA) 49
5.2.4 OASIS XML Common Biometric Format (XCBF) 52
5.3 Integrity/Non-Repudiation 55

5.3.1 Web Services Secure Conversation Language (WS-SecureConversation) 55
5.4 Confidentiality 57
5.5 Trust 58
5.5.1 W3C Signature 58
5.5.2 WS-Trust (GXA) 58
5.6 Authorization/Policy 61
5.6.1 OASIS eXtensible Access Control Markup Language (XACML) 61
5.6.2 W3C Open Digital Rights Language (ODRL) 64
5.6.3 OASIS eXtensible Rights Markup Language (XrML) 67
5.6.4 Web Services Policy Framework (WS-Policy) 69
5.7 General Recommendations 72
5.8 References 72
6 Interoperability of Web Services 75
6.1 Specification and Status 75
6.2 Main Concepts 75
6.3 Recommendation 76
7 Web Services Choreography and Coordination 77
7.1 W3C Web Services Choreography 77
7.1.1 Web Services Choreography Concepts 77
7.1.2 W3C Web Services Choreography Interface (WSCI) 78
7.1.3 OASIS Web Services Business Process Execution Language (WS BPEL) 81
7.1.4 Web Services Transaction (WS-Transaction)/ Web Services Coordination (WS-Coordination) 84
7.1.5 OASIS Web Services Composite Application Framework (WS-CAF) ... 85
7.2 General Recommendations 88
7.3 References 89
8 Web Services and Discovery 90
8.1 Universal Description, Discovery, and Integration (UDDI) 90
8.1.1 Specification and Status 90
8.1.2 Main Concepts 90
8.1.3 Assessment 92

8.1.4 Implementations 93
8.1.5 Recommendation 93
8.2 ebXML Registry 93
8.2.1 Specification and Status 93
8.2.2 Main Concepts 94
8.2.3 Assessment 95
8.2.4 Implementations 95
8.2.5 Recommendation 95
8.3 General Recommendations 96
8.4 References 96
9 The Semantic Web 98
9.1 W3C Web Ontology Language (OWL) 98
9.1.1 Specification and Status 99
9.1.2 Main Concepts 99
9.1.3 Implementations 99
9.1.4 Recommendation 99
9.2 DARPA Agent Markup Language – Semantic (DAML-S) 100
9.2.1 Specification and Status 100
9.2.2 Main Concepts 100
9.2.3 Implementations 101
9.2.4 Recommendation 101
9.3 Topic Maps 101
9.3.1 Specification and Status 101
9.3.2 Main Concepts 101
9.3.3 Implementations 102
9.3.4 Recommendation 102
10 Web Services Monitoring and Management 104
10.1 Specification and Status 104
10.2 Main Concepts 104
10.3 Recommendation 105
11 Applications of Web Services 106
11.1 OASIS Web Services for Remote Portlets 106
11.1.1 Specification and Status 106

11.1.2 Main Concepts 106
11.1.3 Assessment 107
11.1.4 Implementations 107
11.1.5 Recommendation 107
11.1.6 References 108
11.2 Geospatial Web Services 108
11.2.1 Specification and Status 108
11.2.2 Main Concepts 109
11.2.3 Implementations 110
11.2.4 Recommendation 110
11.3 Sensor Web Services 110
11.3.1 Specification and Status 110
11.3.2 Main Concepts 110
11.3.3 Implementations 111
11.3.4 Recommendation 111
12 Preliminary Conclusions and Recommendations 112
Appendix A – Referenced Online Content 116

List of Tables

Table 3.1 Assessment of W3C Web Services Architecture 21
Table 3.2 GXA Specifications 23
Table 4.1 Assessment of WS-Routing/WS-Referral 35
Table 5.1 Security Functionality Categories 37
Table 5.2 Assessment of WS-Security 40
Table 5.3 Assessment of SAML 45
Table 5.4 Assessment of Liberty Alliance 48
Table 5.5 Assessment of WS-Federation 52
Table 5.6 Assessment of XCBF 54
Table 5.7 Assessment of WS-SecureConversation 57
Table 5.8 Assessment of WS-Trust 60
Table 5.9 Assessment of XACML 64
Table 5.10 Assessment of ODRL 66
Table 5.11 Assessment of XrML 68
Table 5.12 Assessment of WS-Policy 71
Table 7.1 Assessment of WSCI 80
Table 7.2 Assessment of WS BPEL 83
Table 7.3 Assessment of WS-Transaction/WS-Coordination 85
Table 7.4 Assessment of WS-CAF 87
Table 8.1 Assessment of UDDI 92
Table 8.2 Assessment of ebXML Registry 95
Table 11.1 Assessment of WSRP 107

List of Figures

Figure 2.1 The SOAP Message Model 10
Figure 2.2 The Main Components of a WSDL Description 12
Figure 3.1 One-Way Message Exchange Pattern 14
Figure 3.2 Request/Response Message Exchange Pattern 15
Figure 3.3 Solicit Response Message Exchange Pattern 15
Figure 3.4 Notification Message Exchange Pattern 15
Figure 3.5 Publish/Find/Bind Triangle 17
Figure 3.6 Enterprise Service Bus 18
Figure 3.7 W3C Web Services Architecture 20
Figure 3.8 ebXML Conceptual Overview 26
Figure 5.1 WS-Security Context Participants 39
Figure 5.2 The SAML Domain Model 43
Figure 5.3 Concept of Single Sign-on 44
Figure 5.4 Federated Network Identity and Circles of Trust 47
Figure 5.5 Federated Network Identity and Circles of Trust 47
Figure 5.6 WS-Federation Model 50
Figure 5.7 Using Security Tokens in WS-Federation 51
Figure 5.8 Use of Trust Engines in WS-Trust 59
Figure 5.9 WS-Trust Interactions 60
Figure 5.10 ODRL Foundational Model 65
Figure 5.11 XrML Concepts and Relationships 68
Figure 7.1 WSCI Interfaces and Collaboration 78
Figure 7.2 WSCI Placement in the Web Services “Stack” 79
Figure 7.3 Example of BPEL4WS Process 81
Figure 7.4 WS-CAF Specifications by Domain 87
Figure 8.1 UDDI Core Data Structures 92
Figure 9.1 The Top Levels of the Service Ontology 100
Figure 11.1 WSRP Mechanism for Aggregating Portlets using Proxies 107
Figure 11.2 O&M Observation Object Model 111

Executive Summary

ES.1 Purpose and Scope

This report represents the analysis of current Web Services specifications, standards, and proposed standards emerging primarily from commercial industry consortiums, with a focus on standards that are relevant to the development of next generation DoD Command and Control (C2) systems.

ES.2 Major Findings

- "Base" Web Services Standards

  The "base" Web Services standards (such as Web Services Definition Language—WSDL—and Simple Object Access Protocol—SOAP) are advancing within W3C. These standards are being adopted broadly but still show some immaturity. The latest versions of WSDL and SOAP have not been widely implemented. Currently these standards don't guarantee unambiguous interoperability.

- Web Services Architectures/Frameworks

  There are multiple efforts to define concrete Web Services architectures. The major effort in this area is the W3C Web Services Architecture.

  The Global XML Web Services Architecture (GXA) specifications are being advanced into open standards consortiums slowly. Only WS-Security has been transferred into an open standards consortium (OASIS).

- Web Services Security

  The current lack of overall robust security makes it difficult to execute Web Services scenarios that stretch beyond "point-to-point" interactions. Point-to-point Web Service interactions using established mechanisms such as traditional Public Key Infrastructure (PKI) and Secure Socket Layer/Transport Layer Security (SSL/TLS) are well established, however.

  A large number of potential standards in the realm of security for Web Services are emerging and will mature in the next two years. The emerging OASIS WS-Security specification will have the largest single impact, and its advancement to OASIS standard level is imminent.

- Web Services Choreography and Coordination

  The choreography and coordination of Web Services can enable inter-organization and inter-agency collaborations. The W3C Web Services Choreography Working Group and the OASIS Web Services Business Process Execution Language (WS BPEL) Technical Committee are beginning to explore this area, but there is a long way to go. Advancements in this area will enable inter-organization and inter-agency collaborations of Web Services.

- Web Services and Discovery

  The current UDDI specifications are still immature and are not specific enough to guarantee portability or interoperability. The weaknesses can be overcome by

limiting the use of UDDI implementations to only standard features and augmenting the UDDI usage by adopting standard practices to achieve the goals for discovery.

The use of service-oriented architectures requires efficient mechanisms by which to discover Web Services descriptions, such as WSDL documents. UDDI serves an important purpose in this regard for both DISA and the federal government, and its adoption is currently on the rise.

The adoption of ebXML Registry has in general been low. However, as adoption of XML grows both within DISA and the federal government, and as more and more XML Schemas are created, the need for an XML registry (such as ebXML Registry) will increase correspondingly.

Both ebXML Registry and UDDI will co-exist and can work together.

- Web Services and Reliable Messaging

  The only open standard that addresses reliable messaging for Web Services is the ebXML Messaging Service (ebMS) specification, but there has not been widespread adoption of this standard. An OASIS Technical Committee (Web Services Reliable Messaging—WSRM) is addressing an open standard for reliable messaging.

- Web Services Interoperability

  The Web Services Interoperability Organization (WS-I) is defining how to achieve interoperability between Web Services standards, but there are few vendor implementations. Guidelines and specifications like WS-I will go a long way toward supporting multi-vendor interoperability by removing the specification ambiguities. We anticipate wide-spread adoption of WS-I.

- Semantic Web Services

  The area of Semantic Web Services is producing specifications (such as the OWL Web Service Ontology Language, OWL-S) that are currently in the research stages. The W3C is a strong proponent of OWL as part of its Semantic Web vision. There is a need for something like OWL, for instance, in discovery services.

- Web Services Monitoring and Management

  There are no released specifications in this important area. There is community interest and at least one early proposal.

- Standards Related to Applications of Web Services

  - OASIS Web Services for Remote Portlets (WSRP) deliver aggregated content to a centralized location. A fair number of WSRP implementations are available.
  - The OGC, an international consortium with significant DoD involvement, has developed a number of Web Services standards for geospatial data sharing, processing, and display. These standards will play a key role in promoting interoperability between C2 systems.

ES.3 Recommendations

- "Base" Web Services Standards

  The "base" Web Services standards (such as Web Services Definition Language—WSDL—and Simple Object Access Protocol—SOAP) should be treated as immature. They should not be regarded as guaranteeing unambiguous interoperability. Organizations such as the DoD, or more likely sub-communities within the DoD, should set forth interoperability guidelines and examples to ensure developers that use this technology use it consistently.

  HTTP(S) and XML should be used for Web Services. XML Schema should be used instead of DTDs.

- Web Services Architectures/Frameworks

  The W3C Web Services Architecture should be watched closely.

  The Global XML Web Services Architecture (GXA) specifications should be watched because of the wide range of functionality that they would cover, if they were transferred into an open standards consortium.

  DISA should not adopt the ebXML framework as a whole, but should only consider individual specifications such as ebXML Registry.

- Web Services Security

  DISA should utilize Web Services at this time, but only in point-to-point interactions using established mechanisms such as traditional Public Key Infrastructure (PKI) and Secure Socket Layer/Transport Layer Security (SSL/TLS).

  OASIS Security Assertion Markup Language (SAML) should be used at this time.

- Web Services Choreography and Coordination

  The work of the W3C Web Services Choreography Working Group bears close watching, as does the emerging work of the OASIS Web Services Business Process Execution Language (WS BPEL) Technical Committee.

- Web Services and Discovery

  The use of UDDI implementations should be limited to only standard features. The use of UDDI should be accompanied by adopting standard practices to achieve the goals for discovery.

  In the future, some of the emerging ontology standards (e.g., OWL), should be used to allow searches based on concepts rather than on specific terms that must now be matched exactly.

  DISA should use UDDI Version 2.0 specification implementations for the time being for its Web Services efforts, and upgrade to Version 3.0 when that becomes an OASIS standard and when an acceptable number of implementations are available.

Once the OASIS/ebXML Registry Version 2.5 specifications reach a Version 3.0 status, an assessment should be made regarding available implementations and consideration should be given to implementing ebXML Registry.

- Web Services and Reliable Messaging

  The area of reliable messaging should be watched to see which specification emerges as the leader from the current ones.

- Web Services Interoperability

  DISA should hold off from utilizing any of the Web Services Interoperability Organization (WS-I) profiles (such as the WS-Basic Profile) until more vendor implementations emerge.

- Semantic Web Services

  DISA should watch the area of Semantic Web Services for ontology language standards.

- Web Services Monitoring and Management

  DISA evaluate the OASIS Web Services Distributed Management (WSDM) specifications once they are released.

- Standards Related to Applications of Web Services

  Although a number of OASIS Web Services for Remote Portlets (WSRP) implementations are available, DISA should wait about 6 months before recommending widespread use in the DoD.

- Geospatial Web Services

  DISA should use OGC standards for promoting interoperability between C2 systems, as well as for supporting competitive procurement.

ES.4 Conclusions

In general, Web Services are usable and useful today, but implementers must get past the general myth that the current Web Services standards guarantee interoperability. Interoperability is enhanced by these standards, mainly through simplification, such as using ubiquitous communications channels (HTTP), and a simplified, man-readable, yet structured and flexible data format (XML). However, the key to interoperability is the semantics of the connection. Additional standards including XML Schema, SOAP, and WSDL also add value, but still do not clearly convey the semantics without human intervention. To solve the problem additional specifications (including application-specific specifications) need to be adopted.

The DoD should continue and expand upon current efforts to use Web Services standards as an alternative to traditional means of creating systems. Providing a more open environment to support access to the services and data of C2 systems will foster new and more creative solutions which can leverage a wide array of sensors and databases.

1 Introduction

This report presents the results of an analysis of Web Service standards that may support DoD requirements. It is the first in a series of papers that will also include:

- Web Service Standard C2 User Requirements
- Emerging Web Services Development Environment

This overall effort involves analysis of existing and emerging (proposed) standards supporting Web Services, and evaluates the potential impact on DoD Command and Control (C2), in order to:

(1) Influence use of commercial standards to promote DoD interests

(2) Develop and convey an understanding of Web Services standards issues from a variety of Web Services standard organizations; and

(3) Disseminate timely information concerning commercial standards to DoD users.

1.1 Conventions

As with any written document, this report only represents a snapshot of the specifications, standards, consortium documents, and vendor products. It was the intent of the analysts to cover all technologies related to the implementation of Web Services in the C2 domain. However, this does not imply that all specifications and products were found and/or recognized to be relevant; in other words, some relevant specifications or products may have been unintentionally omitted. Others may soon appear as new or now relevant, but were not in time to be included in this document.

In this report, specifications are described in detail if the majority of the industry currently recognizes them as official standards, or if they appear to be clearly headed toward standardization in order to fill a known gap in the successful and broad application of Web Services.

Some vendor implementations of the standards are also listed for the convenience of the reader. Readers are strongly advised to consider both the inevitable biases of individuals evaluating these products, and the changing landscape where new and/or upgraded products are constantly emerging. The products listed likely do not represent the only, or even the best implementations of the specifications. A product is included if it has strong market presence and possibly (if found) good reviews relative to the implementation of the standard.

1.2 How to Read this Document

This report is both an overview of the current Web Services landscape and a reference for technologists that want to gain a thorough understanding of the technologies and how they apply. It is not meant to be read continuously, cover-to-cover. A recommended reading pattern is to start with the Section Overview, below, to select interest areas; then, for each section and subsection of interest, read the introductory paragraphs at the beginning, and the Recommendations at the end of each subsection. At this point, the final section on Conclusions and Recommendations will be easier to follow. Reading the remaining detail may be left to only those subsections of strong technical interest.

1.3 Section Overview

This Analysis of Web Services Standards presents the following major sections:

Section 2 Web Services

This section introduces the concept of Web Services; the foundational specifications that support this concept, such as XML, SOAP, WSDL, and UDDI; and the standards consortiums involved in the current evolution of Web Services and standards.

Section 3 Web Services Architectures

This section discusses various distributed application architectures to which Web Services can be applied; how Web Services are used in the implementation of these architectures; and some specific Web Service architecture specifications being adopted.

Section 4 Web Services and Messaging

This section describes the specific transfer protocols, message formats, and challenges being addressed—such as performance—related to the transfer of XML messages to support Web Services.

Section 5 Web Services and Security

This section discusses a number of topic areas related to the application of security to Web Services in order to attempt to ensure safe and secure communication between interacting parties. These topics include:

- Authentication
- Identity Management
- Integrity
- Confidentiality
- Authorization
- Non-repudiation
- Trust
- Policy

Section 6 Interoperability of Web Services

This section covers one of the major issues in implementing a distributed communication protocol using open standards, with technology and software from multiple vendors: ensuring interoperability. One of the most important reasons for the popularity of Web Services is that it encourages automation-based communication between disparate organizations, each of which may have different IT strategies, products, vendor relationships, and technical expertise. Web Services are meant to remove most of the barriers to electronic business relationships, but this can only succeed if the protocols used for communication are so well defined that they can guarantee interpretation on each end of the connection.

Section 7 Web Services Choreography, Workflow, Mediation, and Routing

This section discusses the process of defining and organizing complex transactions involving multiple entities (including but not limited to Web Services). Web Service standards are naturally being augmented by standards for modeling and controlling business processes. This section describes some of the emerging specifications and proposed standards.

Section 8 Discovery of Web Services

This section describes the applications of registries to support automated and semi-automated discovery of and connection to Web Services over a network.

Section 9 The Semantic Web

This section discusses semantic technology being applied to Web Services to support discovery, automated connection, and even automated reasoning about the operations and message content in Web Services. Some of these capabilities are not likely to see practical use in the near term, but others have well-understood, useful applications to meet important needs of Web Service architectures.

Section 10 Web Services Monitoring and Management

This section discusses the need for monitoring and management of Web Services, and some emerging and proposed specifications to support this as-yet-unaddressed need.

Section 11 Applications of Web Services

This section discusses some common applications of Web Services, and some specifications on the periphery of Web Services standards, that are relevant to DoD C2 net-centric software development.

Section 12 Preliminary Conclusions and Recommendations

This section is called “preliminary” because it summarizes the initial conclusions and recommendations drawn from this Web Services analysis phase of the Web Services Standards Analysis task. As the remaining tasks and resulting documents are produced, a final set of conclusions and recommendations will be written taking into account the C2 user requirements and the emerging Web Services development environment.

2 Web Services

The notion of a 'Web Service' has proven to be difficult to define. The W3C Web Services Architecture group has debated various candidate definitions vigorously, for many months, without reaching a consensus. (Rather than consensus, their current definition appears to reflect a lack of willingness to continue the debate, coming from all sides, which seems to be holding. See: http://lists.w3.org/Archives/Public/www-ws-arch/2003Aug/0047.html)

The major candidates definitions involved in the definition debate are (1) a Web Service is anything that passes XML messages over HTTP, or (2) anything that could be defined with WSDL, or (3) anything that is described by WSDL and uses SOAP. Some of the major issues involved in the debate have been (a) whether support for the higher-level infrastructure services defined by the Web Services architecture (security, reliability, etc) was provided by the terms of the definition, (b) whether CORBA, DCOM, and their ilk should be excluded by the definition, and (c) whether web servers and browsers should be excluded by the definition. Opinions about the importance of issues such as (a), (b), and (c) seemed to influence preferences for definition (1), (2) or (3).

2.1 Web Services defined

The W3C architecture group says the following:

> There are many things that might be called "Web Services" in the world at large. However, for the purpose of this Working Group and this architecture, and without prejudice toward other definitions, we will use the following definition:
>
> Definition: A Web Service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web Service in a manner prescribed by its description using SOAP-messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards.

One useful way to read this definition is a profile of the kind of Web Services that the architecture group is interested in. And under this reading it becomes possible to consider other characteristics of Web Services, besides WSDL and SOAP that do not logically follow from the fact of being defined in WSDL, or from the fact of using SOAP. Such further characteristics are discussed in the next section.

2.2 Characteristics of Web Services

Almost every discussion of Web Services lists a number of characteristics that Web Services have, or could have, or perhaps should have. A quick survey of these discussions based on a Google search shows very little overlap in the sets of characteristics that have been suggested. Frequently the suggested characteristics are qualified as being "defining" or as being "essential" characteristics; this is problematic, given the evident lack of overlap. Further, there is normally very little indication of whether the suggested characteristics are characteristics of existing Web Services, of Web Services as they are or would be defined in some version of Web Services architecture, or of something else.

The most commonly cited characteristic of Web Services is that they are loosely coupled. But some careful thought about this characteristic has shown that it is actually a combination of other properties, any of which may be present to a greater or lesser degree. (See: http://lists.w3.org/Archives/Public/www-ws-arch/2003Sep/0086.html and comments.) If this is the case, then does loosely coupled still count as a characteristic?

In order to better understand the nature and importance of the characteristics of Web Services, this report adopts the following framework for discussion. First, the characteristics of an ideal middleware are laid out, and then Web Services are compared to this ideal middleware, in order to see which of the characteristics they inherit.

2.2.1 The Ideal Middleware

The notion of an ideal middleware comes from the work of B. Benatallah and his colleagues at the University of New South Wales. (See, for example: http://www.sistm.unsw.edu.au/people/rabhi/publications/booksection.pdf).

The central concept in this discussion is integration, which provides interoperability to heterogeneous systems and applications (the integration "partners").

Integration occurs at different layers, and the ideal middleware provides seamless integration at all of them. The integration layers are:

- **The Communications Layer:** This is the level at which information in exchanged between the partners. Information exchange can be in the form of messages, or remote procedure calls to distributed objects. For this layer the ideal middleware would completely hide the protocol and/or distributed object framework details from the partners.
- **The Content Layer:** This is the layer at which concepts and their properties get mapped into data formats, and then get read back out again. For this layer the ideal middleware would provide complete semantic and syntactic interoperability, by hiding the transformations of data formats and the translations of concepts from the partners.
- **The Business Process Layer:** This is the layer at which joint business processes are carried out. Workflow to workflow interactions would be an example. For this layer the ideal middleware would provide transparent peer-to-peer business carried out with arbitrary partners.

The characteristics of an ideal middleware are then these: protocol hiding, distributed object framework hiding, data format hiding, hiding the translation of concepts into data formats and back out again, and hiding the details of inter-enterprise business process integration. Each of these characteristics occurs at a certain level.

2.2.2 Web Services as Ideal Middleware

By almost any definition, Web Services exchange XML over HTTP. Just this much gives Web Services some of the characteristics of ideal middleware, in the communications layer.

Because XML is text based, it hides low-level details of data encoding, such a big-endian vs. little-endian, character set encoding, and number representation. The use of standard XML Schema data types allows Web Services to go even further in this direction. XML also provides extensibility to data formats, in the sense that formats can be changed by the addition of new fields and attributes, without affecting (or "breaking") clients that are expecting to see the old format.

HTTP (plus DNS) hides the details of the physical location of services, allowing symbolic references with URLs. And the HTTP protocol itself hides lower level protocol details, such as making and tearing down connections, reliable transport, and (with SSL) secure channels between HTTP nodes.

When Web Services use SOAP they acquire additional characteristics of ideal middleware. SOAP provides a standard form of distributed error handling, and thus hides the details of an important, though often ignored, area. SOAP with attachments provides a standard way of assembling messages that have parts, and thus provides one sort of interoperability at the content level. And SOAP headers provide the hooks for reliable messaging and data security. Insofar as these can be automated and hidden from the business process layer, they become characteristics of ideal middleware.

Interoperability at the content layer is perhaps the least-addressed part of ideal middleware, from the Web Services point of view. Shared XML Schemas partially addresses the goal of hiding the details of data format conversions, but the process of developing such schemas requires all partners to look closely at details, and at how those details do or don't fit into the shared schema. Shared ontologies partially address the goal of hiding the translating of concepts into and out of (shared) data formats.

Interoperability at the business process layer is certainly being addressed by Web Services standardization efforts, which are building on XML standardization efforts such as ebXML or RossettaNet, which are themselves building on earlier EDI efforts. But this level of interoperability is not characteristic of existing Web Services. Rather, the clear goal is for it to be a characteristic of Web Services as they will exist in the future.

2.3 Foundations of Web Services

2.3.1 Extensible Markup Language (XML)

The Extensible Markup Language (XML) is a subset of SGML that can be served, received, and processed on the Web in the way that HTML can.

2.3.1.1 Specification and Status

Extensible Markup Language (XML) 1.0 (Second Edition) W3C Recommendation 6 October 2000

XML 1.0 Second Edition Specification Errata (as of 2003-09-10)

2.3.1.2 Main Concepts

Documents vs. Data

XML was designed to enable electronic publishing on the Internet. Within the publishing industry, the tradition from the beginning has been for editors to insert so-called "markup" symbols into a text, in order to tell typesetters how to set up a page of type. One of the real advances of XML was to define a way of inserting markup that allowed marked-up text to be parsed unambiguously.

Once XML was under development, it quickly became evident to a number of people that XML had another use. Rather than inserting the markup into the content, as publishers do, one could instead insert content into the markup. XML markup allowed unambiguous parse trees to be defined, and these trees could be used to "hold" the content. People working on formal product descriptions, such as are found in manufacturers' catalogs, were among the first to see the potential for this structured content use of XML.

For Web Services purposes it is the structured content use of XML that is important. Web Services need to ingest and react to messages without human intervention, and for that to happen those messages need to have unambiguous structure. Further, the useful function of many Web Services is to respond to queries for data, and the structured content use of XML has turned out to be a good way to provide that function.

XML Styles

Given that XML is to be used to provide unambiguous structure, there are three predominant styles for how that structuring is done. These can be used independently or in combination. There is an interaction between the style of an XML document and the schematic description of the XML used.

The first style of structuring attempts to copy some inherent structure of the data that is to be represented into the structure that is provided by the XML tagging. This style tends to work best with things that are designed to be structured; a typical example is a bibliographic record. A bibliographic record has a designed-in structure where a book has a title and an author, and author has a first name and a last name, and so on. Structures like these can often be imitated exactly by XML.

The second style of structuring uses some linking mechanism that is provided by XML or by an ancillary technology such as XLink. In this case, the inherent structure of the data being represented is modeled by relationships that are created with links between elements that are, from the point of view of the first style, independent. Many good examples of this style can be found by looking at XML encoded RDF documents.

The third style resembles the second, except that some ad-hoc mechanism is used to create the links that establish the relationships betweens the parts of the structure. WSDL documents are a good example of this style. In WSDL documents links are created using the principle of identical names within a namespace. There is nothing in the structure of XML itself that says that this principle should hold. It is just a convention that is established outside of XML, by the WSDL specification itself.

2.3.1.3 Implementations

A large and growing number of implementations are available, without cost.

2.3.1.4 Recommendation

Level 1: Ready for use.

Despite its short history, XML already permeates the web, both in terms of domains and in terms of geography. (Source: The XML Web, http://www2003.org/cdrom/papers/refereed/p677/p677-mignet.html)

2.3.2 XML Schema

From the beginning XML had a DTD language that allowed the description of the general form of a type of structure. Here "general form of a type" means that a structure could have variants and could still be described. This ability to describe general forms is useful in its own right; the DTD language can be used not only to describe actual data, but also to specify how potential data sets are required to be structured. However, the DTD language is not well suited for describing messages and data, because it lacks namespaces and actual data types

XML Schema is a more powerful language for describing XML structure; it provides a large set of data types that are now in use in a number of the ancillary W3C XML technologies. It provides the ability to define structures and parts of structures within namespaces. In addition, it provides the capability for doing modularized, object-oriented development of schemas for XML. And of course it provides validation.

2.3.2.1 Specification and Status

XML Schema Part 1: Structures (W3C Recommendation 2 May 2001)

http://www.w3.org/TR/xmlschema-1/

XML Schema Part 2: Data types (W3C Recommendation 02 May 2001).

http://www.w3.org/TR/xmlschema-2/

2.3.2.2 Main Concepts

Validation

Validation is the process of checking the conformance of a document to a schema. In the case of XML Schema, checking the conformance of an XML document involves checking structural matches, and checking data types. If the XML structure of a document is such that it mirrors the structure of the underlying model of the document, then validation can be used to check the conformance of a document with its intended model.

Modular and Object-Oriented

XML Schema is designed to support modular object-oriented schema development. Support for modular development is provided by namespaces and import and/or include capabilities. Support for object-oriented development includes control over visibility with

global vs. local class and element declarations, object class inheritance via the use of substitution groups, and extension and/or restrictions of classes.

XML Schema and Web Services

By itself, XML, and the structure that it provides, do not enable the interoperability of Web Services. At least one more level of cooperation between services is required, in order for them to make use of the structured XML messages and data that they exchange. Somehow a shared understanding of the structure of the XML has to be present at both ends of the message channel. XML Schema provides one way of doing this, insofar as it provides a language for specifying what the messages and data have to look like in general.

The intricate process of developing schemas for data model components, and building useful application schemas out of these components (which is so specialized that it even has a name), is critical for interoperability.

2.3.2.3 Implementations

- XML Spy
- Xerces
- XSV

2.3.2.4 Recommendation

Level 2: Widely accepted but difficult to use correctly

The need for the development of best practices for using XML Schema (a need which is currently being fulfilled) is a good indication of the difficulty of developing realistic XML Schema.

2.3.3 Simple Object Access Protocol (SOAP)

SOAP is a lightweight protocol for exchange of information in a decentralized, distributed environment. It is an XML based protocol that defines the following four things:

1. A SOAP message construct that defines the structure of a SOAP message in terms of XML infosets for SOAP envelopes, headers, bodies, and faults.
2. A SOAP processing model that defines the rules for processing a SOAP message
3. A protocol binding framework, that defines the rules for defining a binding to an underlying protocol that can be used for exchanging SOAP messages between SOAP nodes. Special attention is given to the HTTP protocol binding.
4. An extensibility model that defines the concepts of SOAP features and SOAP modules.

2.3.3.1 Specification and Status

SOAP Version 1.2 Part 1: Messaging Framework (W3C Recommendation 24 June 2003)

SOAP Version 1.2 Part 2: Adjuncts (W3C Recommendation 24 June 2003)

2.3.3.2 Main Concepts

Messages

SOAP is fundamentally a stateless, one-way message exchange paradigm. Hence the notion of message is central. Figure 2.1 lays out the conceptual landscape of the SOAP message, with the message at the center.

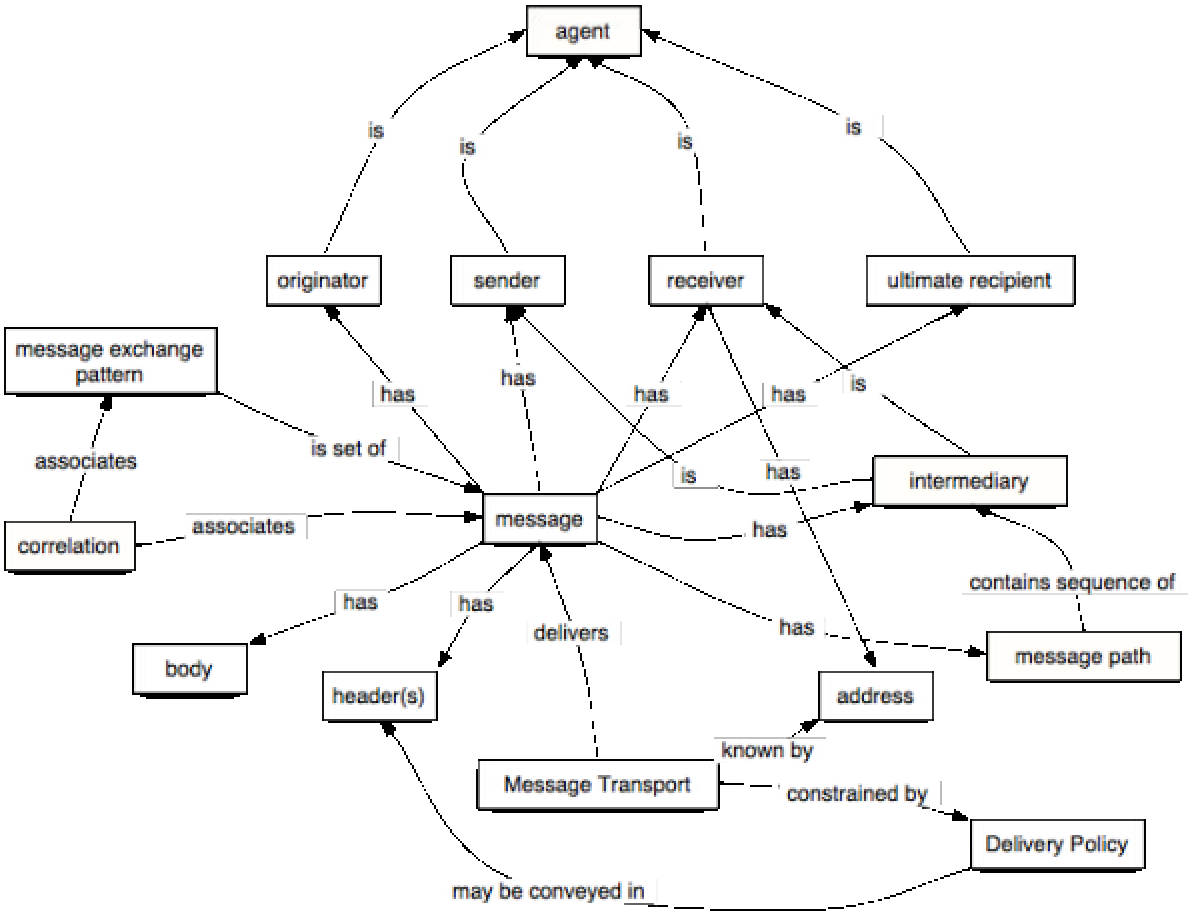

Figure 2.1 The SOAP Message Model

Source: http://lists.w3.org/Archives/Public/www-ws-arch/2003Oct/0015.html

SOAP and Web Services

For the W3C Web Services architecture, SOAP is part of the defining characteristics of a Web Service. This is because it is SOAP that provides the structure that is needed for higher-level requirements, such a reliable messaging and security, to be realized.

2.3.3.3 Implementations

Major vendors and open source groups are committed to SOAP. Initial support for SOAP 1.2 is available in the following implementations:

- Axis

- .NET

2.3.3.4 Recommendation

Level 2: Ready for early adopters.

SOAP 1.1 has been very widely implemented and is part of the WS-I Basic Profile Version 1.0. SOAP 1.2 went to Recommendation status in June, 2003. It does not seem likely that SOAP 1.2 will be particularly controversial and major vendors will probably implement it quickly now that it is a recommendation.

2.3.4 Web Services Description Language (WSDL)

Web Services Description Language (WSDL) provides a model and an XML format for describing Web Services at the operation level. WSDL is way to describe a contract between a consumer and a service about the exchange of messages between them. XML Schema (or a related schema language) is used to describe the structure of those messages.

2.3.4.1 Specification and Status

Web Services Description Language (WSDL) 1.1 (W3C Note 15 March 2001)

Web Services Description Language (WSDL) Version 1.2 Part 1: Core Language (W3C Working Draft 11 June 2003)

Web Services Description Language (WSDL) Version 1.2 Part 2: Message Patterns (W3C Working Draft 11 June 2003)

Web Services Description Language (WSDL) Version 1.2 Part 3: Bindings (W3C Working Draft 11 June 2003)

2.3.4.2 Main Concepts

Abstract Interface Definitions

WSDL distinguishes the abstract description of a service from the concrete details of how to access an actual implementation of a service. The top outlined area in Figure 2.2 shows the WSDL components that contribute to the abstract definition, and the bottom outlined area shows the concrete implementation components. The purpose of this distinction is to be able to define both service types and service instances, and to be able to relate them. (Note that in WSDL version 1.2 ‘portType’ has become ‘interface’ and ‘port’ has become ‘endpoint’.)

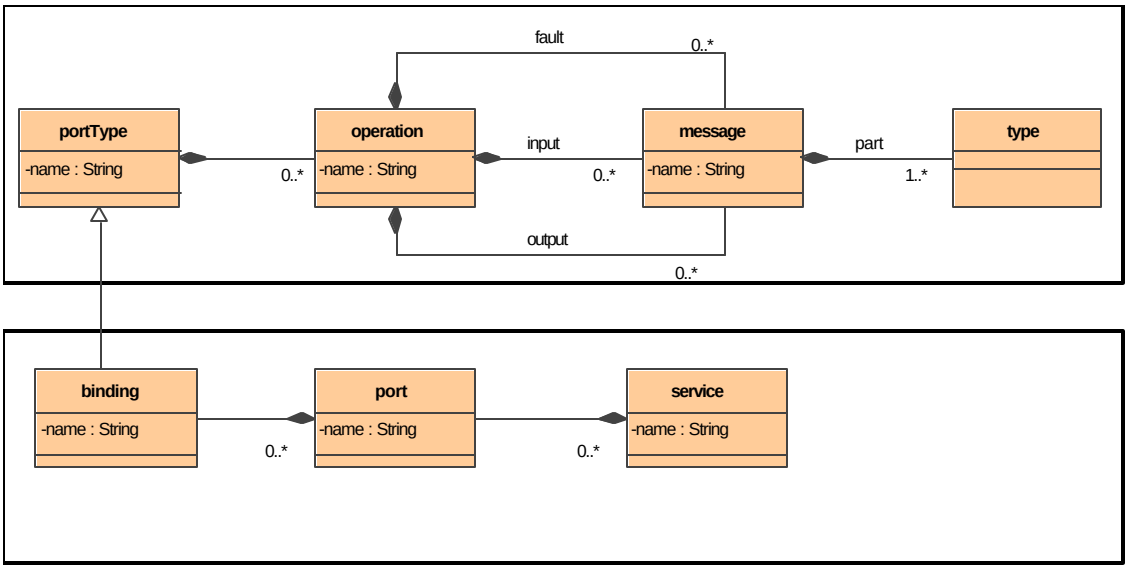

Figure 2.2 The Main Components of a WSDL Description

Message Exchange Patterns

Message exchange patterns are a central feature of WSDL. A message exchange pattern is composed of a set of messages, together with their senders and receivers that make up a single use of a Web Service. The term 'operation', which is used in Figure 2.2, might implicitly suggest that an operation describes something more than a message exchange. In WSDL, that is not the case. That a particular operation gets mapped, in some application, to, say, a method invocation is not suggested or implied by the existence of a WSDL operation component. Rather, what is intended by the term ‘operation’ is a particular level of interaction – the level that focuses on a particular service node – in a stack of increasingly more complex interactions among Web Services. Extremely simple applications based on single message exchanges may be adequately characterized at the operation level.

Bindings

WSDL defines bindings for SOAP, and for HTTP GET and POST methods. Additionally, it provides an extensibility mechanism for defining additional technology specific bindings. This allows for changes in the area of network and message protocols, without requiring coordinated changes in the WSDL specification.

2.3.4.3 Implementations

A number of implementations of tools for developing WSDL 1.1, and applications that use WSDL 1.1 are in existence. Currently the tools do not interoperate, but there is active work that is focused on making that happen. Many of these tools operate on the assumption that WSDL is an interface definition language, something that the authors of version 1.2 are trying to discourage.

- WSDL4J
- .NET
- WSDL2Java

- CapeClear

2.3.4.4 Recommendation

Level 2: Ready for early adopters

Although WSDL 1.1 has never been on track to become an official specification, it has been very widely implemented and is part of the WS-I Basic Profile Version 1.0. WSDL 1.2 is being developed in the W3C and is in a "middle" stage of the standardization process. There does not seem to be any particular competition to WSDL 1.2 in other standards organizations and major vendors will probably implement it quickly once it becomes a recommendation (which will take a while).

3 Web Services Architectures

This section describes architectural considerations for Web Services. Various "architecture types" are described (service-oriented, peer-to-peer), as well as specific architectures/frameworks that are especially noteworthy. This section begins with a discussion of message exchange patterns (MEPs), as they play a vital role in Web Services architectures as well as many of the specifications that we will reference in this document. We offer recommendations only on the specific architectures/frameworks; the remainder is meant as supporting information.

3.1 Message Exchange Patterns

In considering the interactions between a client (requestor) and a Web Service (provider), there are various possible *message exchange patterns (MEPs)* that define these interactions. The W3C Web Services Definition Language (WSDL) Version 1.1 defines the following four message exchange patterns:

- One-way
- Request-response
- Solicit-response
- Notification

The One-way and Request-response patterns are the most widely used in current practice. Each of these patterns is described below.

3.1.1 One-Way

In this message exchange pattern, the Web Service receives a message from the client:

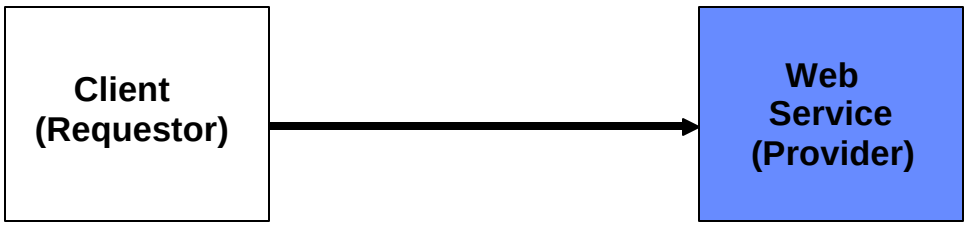

Figure 3.1 One-Way Message Exchange Pattern

An example would be the submission of a purchase order to a Web Service.

3.1.2 Request/Response

In this message exchange pattern, the Web Service receives a message from the client and sends back a correlated message:

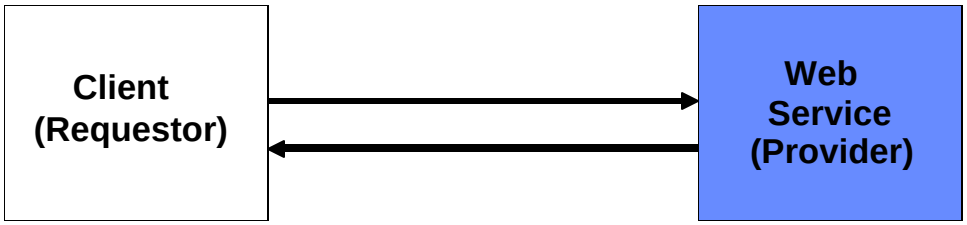

Figure 3.2 Request/Response Message Exchange Pattern

An example would be the submission of a purchase order to a Web Service followed by an acknowledgement from the Web Service to the client.

3.1.3 Solicit Response

In this message exchange pattern, the Web Service sends a message to the client and receives a correlated message:

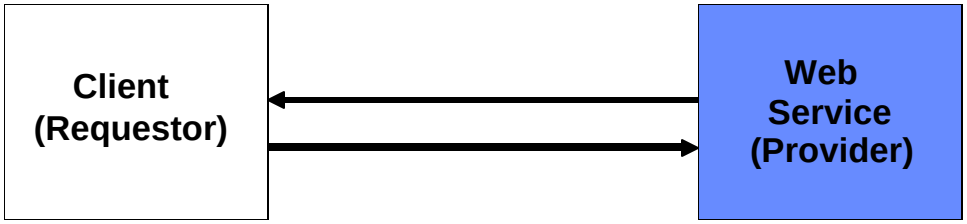

Figure 3.3 Solicit Response Message Exchange Pattern

An example would be the sending of an invoice to a client by Web Service in response to the receipt of a purchase order by the Web Service, in which the Web Service did not receive an acknowledgement of receipt of the invoice by the client within a predetermined period of time. The Web Service would then send a "request for acknowledgement".

3.1.4 Notification

In this message exchange pattern, the Web Service sends a message to the client:

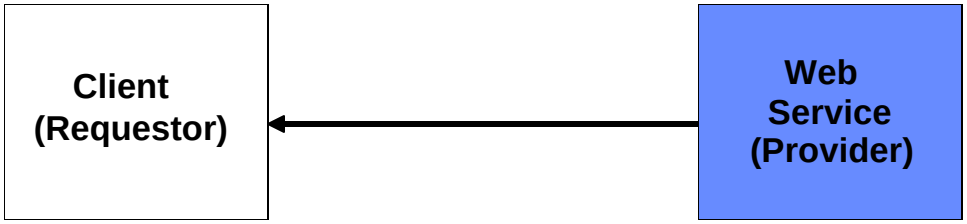

Figure 3.4 Notification Message Exchange Pattern

An example would be the sending of an invoice to a client by Web Service in response to the receipt of a purchase order by the Web Service.

In addition to the message exchange patterns described above, the notion of *synchronization* is also important. In terms of synchronization, there are two types of message exchanges:

- Synchronous
- Asynchronous

In a synchronous message exchange, the client waits for a response from a Web Service after sending a request, and receives a response from the Web Service on the same communication channel on which the request was sent. The process of waiting for a response is known as "blocking". Conversely, in an asynchronous message exchange, the client does not wait for a response from a Web Service after sending a request; rather, the client is free to perform other tasks and receives a response a response from the Web Service at a later point on a different communication channel on which the request was sent. The response message is "correlated" to the corresponding request message through one or more pieces of information in the response (such as a request ID, purchase order number, etc.). Asynchronous message exchanges often utilize message queuing mechanisms such as Java Message Service (JMS) or IBM WebSphere MQ.

Each of the two-message message exchange patterns discussed above (request-response and solicit-response) may be implemented in either a synchronous or asynchronous manner.

3.2 Architecture Types

This section presents various "architecture types" that can be considered for use with Web Services. Of all the architecture types described here, the service-oriented architecture (SOA) is most prominently used with Web Services today.

3.2.1 Service Oriented Architecture (SOA)

A *service-oriented architecture (SOA)* is similar in concept to a distributed system, but it is focused on the concept of services. A distributed system is a system that is comprised of various "components" that do not share a common address space (i.e. common memory) and do not operate in the same processing environment. Distributed systems have existed for many years, and technologies such as Microsoft's Distributed Component Object Model (DCOM) and the Object Management Group's (OMG's) Common Object Request Broker Architecture (CORBA) have been used to facilitate communication between distributed program objects in a distributed system.

A service-oriented architecture is essentially a collection of services that communicate with each other. The communication can involve either simple data passing or it could involve two or more services coordinating some activity. The following concepts are essential to service-oriented architectures:

- **Services:** Referred to as "Web Services" in current SOA architectures
- **Dynamic discovery:** Services can be dynamically discovered through mechanisms such as service registries
- **Messages:** Service providers and consumers communicate via messages

The "Publish/Find/Bind" Triangle

In an SOA, services are *published* in a service registry where consumers (requestors) *find* them. Once found, a consumer *binds to* (uses) a service. This is known as the "publish/find/bind" triangle, and is illustrated in the figure below:

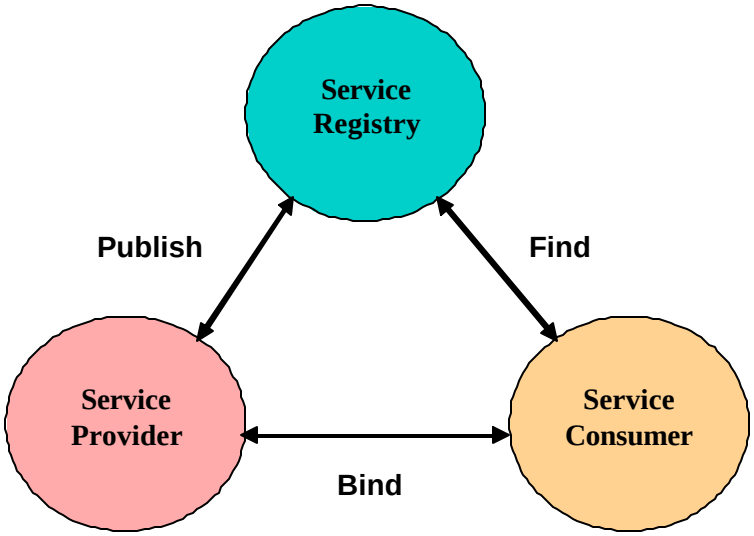

Figure 3.5 Publish/Find/Bind Triangle

3.2.2 Peer-To-Peer (P2P)

In a *peer-to-peer (P2P) architecture*, all nodes are treated as equals (peers) in a decentralized fashion. There is no need for a central server (such as the Service Registry in the SOA figure above), as any node can provide this functionality. Therefore, each node would be considered both a requestor and a provider.

There are some disadvantages to peer-to-peer architectures that render them less attractive than service-oriented architectures:

- **Bandwidth:** Because P2P architectures eliminate central servers, all peers in a network are searched – this uses a large amount of bandwidth.
- **Security:** Increased security risks due use of decentralized services.
- **Maintenance:** Due to its complex architecture, maintenance of a P2P network can be difficult.

3.2.3 Enterprise Bus

Enterprise Service Bus (ESB) is an emerging architecture that is backed by software vendors such as Sonic Software, Cape Clear, and Tibco. IBM recently announced plans for releasing an ESB product. As with P2P architectures, all applications and services in an ESB are viewed equally as service endpoints. An ESB is a type of SOA in which communications between endpoints are asynchronous.

The following figure illustrates the concept of an ESB:

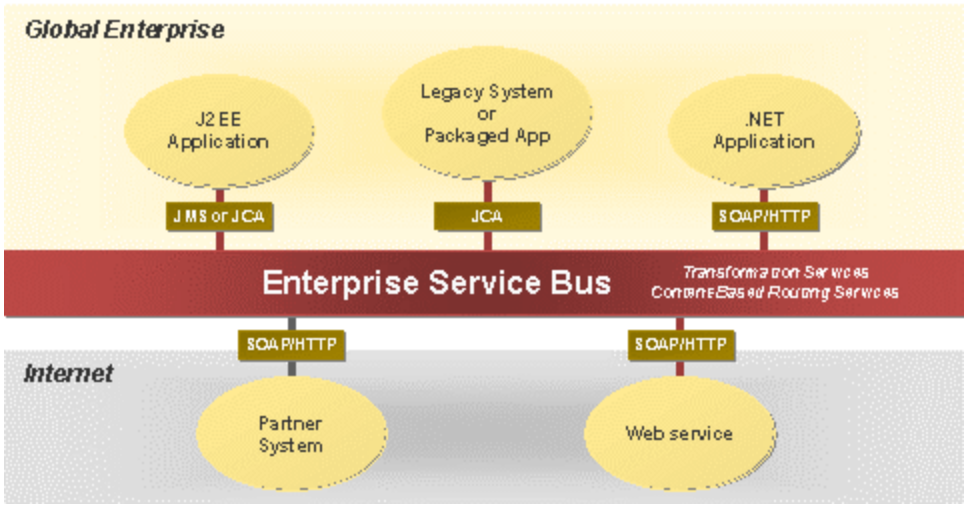

Figure 3.6 Enterprise Service Bus

Source: WebServices.org

In the above figure, applications and services simply “plug into” the ESB, post their data to the ESB, and receive data from it. The ESB takes care of the necessary coordination between interactions.

3.2.4 Grid Computing

Grid computing involves applying the resources of many computers in a network to a single problem at the same time, thereby making more cost-effective use of a given amount of computer resources. This is usually applied to a scientific or technical problem that requires a great number of computer processing cycles or access to large amounts of data. An initiative known as the Open Grid Services Architecture (OGSA) represents an evolution towards a grid system architecture based on Web Services concepts and technologies. OGSA is built on a base infrastructure known as Open Grid Services Infrastructure (OGSI). OGSI’s "Grid Service Specification" defines the standard interfaces and behaviors of a Grid service, building on a Web Services base.

The Globus Alliance is a research and development project focused on enabling the application of grid concepts to scientific and engineering computing. Its major corporate partners currently include IBM and Microsoft. Hewlett-Packard Company has announced plans to further enable its enterprise infrastructure technologies for grid computing by incorporating support for the Open Grid Services Architecture (OGSA) and Globus Toolkit (an open-source software toolkit for building grids) into its product lines.

3.3 Specific Architectures

This section presents various specific emerging Web Services architectures and frameworks. Although ebXML is not considered to be a Web Services architecture or framework, it is included because of its various Web Services aspects.

3.3.1 W3C Web Services Architecture (WSA)

The W3C Web Services Architecture (WSA) Working Group was initiated in January 2002 as part of the W3C Web Services Activity. The goal of the W3C Web Services Activity is to develop a set of technologies in order to lead Web Services to their full potential. The expected duration of the W3C Web Services Architecture Working Group is 2 years, through January 2004. The W3C Web Services Architecture is a W3C Working Draft, dated August 2003.

The W3C Web Services Architecture integrates different conceptions of Web Services under a common "reference architecture". In doing so, it provides a model and a context for understanding Web Services, and the relationships between the various specifications and technologies that comprise the Web Services Architecture. Additionally, the W3C Web Services Architecture describes both the minimal characteristics that are common to all Web Services, and a number of characteristics that are needed by many, but not all, Web Services.

3.3.1.1 Specification and Status

This section discusses the W3C Web Services Architecture Working Draft, August 2003.

3.3.1.2 Main Concepts

Web Services “Stack”

The W3C Web Services Architecture defines the following “stack diagram” for Web Services:

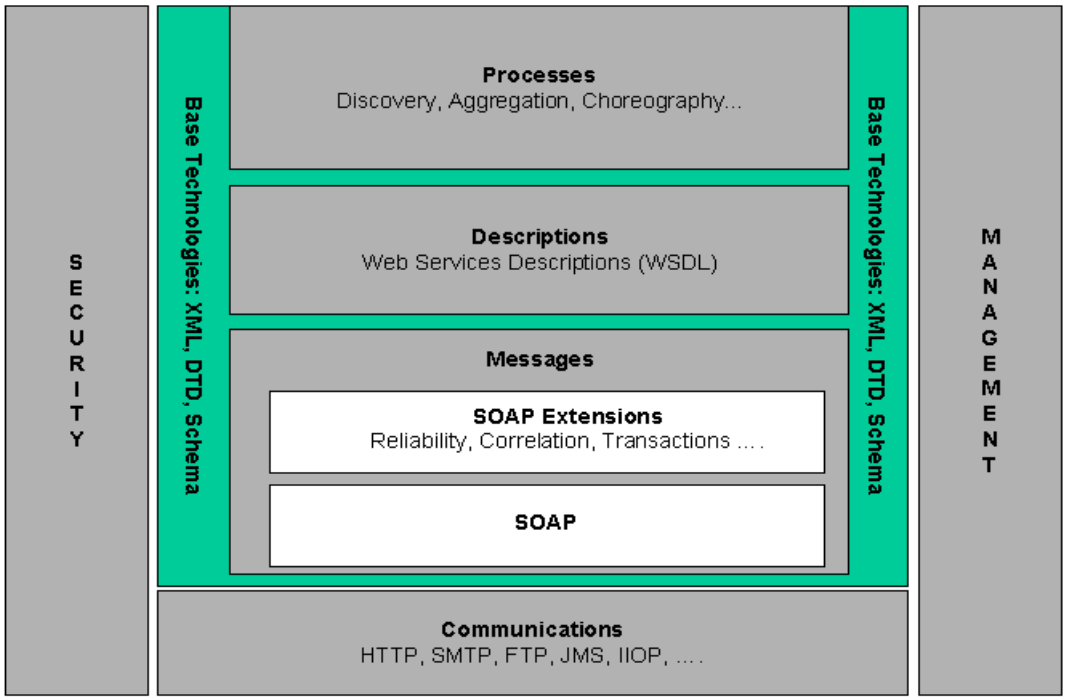

Figure 3.7 W3C Web Services Architecture

Source: W3C

This diagram ranges from various communications/transfer protocols at the bottom (HTTP, SMTP, etc.), moving upward through the SOAP messaging layer, Web Services descriptions, and arriving at the "Processes" layer which encompasses areas such as Web Services discovery and Web Services choreography. The stack also conveys the high importance of security and management through their vertical placement spanning all other layers.

Architecture Models

The W3C Web Services Architecture consists of five "architecture models":

- Message-Oriented Model (MOM)
- Service-Oriented Model (SOM)
- Resource-Oriented Model (ROM)
- Policy Model
- Management Model

Each model provides a different "lens" through which to view Web Services. Each is described in further detail below:

- The **Message-Oriented Model (MOM)** focuses on those aspects of the architecture that relate to messages and their processing. It addresses how Web Service agents (requesters and providers) may interact with each other using a

message oriented-communication model in which XML-formatted messages are exchanged.

- The **Service-Oriented Model (SOM)** builds on the MOM by adding the concept of services and actions that are performed by service requesters and service providers. The SOM essentially allows us to interpret messages as requests for actions and as responses to those requests.
- The **Resource-Oriented Model (ROM)** focuses on those aspects of the architecture that relate to resources (i.e. anything that has an identifier), and the service model associated with manipulating resources. It builds on the SOM through its development of the service model associated with resources, and common actions associated with manipulating resources.
- The **Policy Model** focuses on the core concepts needed to relate policies to Web Services. Policies may be enacted to represent security concerns, quality of service concerns, management concerns and even application concerns.
- The **Management Model** focuses on the management of Web Services, including use of the infrastructure offered by Web Services to manage the resources needed to deliver that infrastructure. The Management model uses many of the other features and concepts of the architecture, such as the concepts of resource, description, service, etc. It also addresses the life cycle of Web Services.

The REST Architectural Style

REST ("Representational State Transfer") is an architectural style for Web applications that is referenced with the W3C Web Services Architecture. Proposed by Dr. Roy Fielding, it is a technique in which agents manipulate only by the exchange of "representations", thereby using "hypermedia as the engine of application state." A simpler alternative to SOAP-based Web Services, REST defines identifiable resources, and methods for accessing and manipulating the state of those resources. As implemented on the World Wide Web, URIs identify the resources, and HTTP is the protocol by which resources are accessed. Unlike RPC-based techniques such as SOAP that use their own syntax for describing the location of information, REST uses HTTP and its PUT, GET, POST and DELETE operators for the exchange of data.

REST is favored by some for Web Services because of its simplicity; however, its limited set of operators and its "low-level" interface lead others to prefer a SOAP-based approach.

3.3.1.3 Assessment

Table 3.1 Assessment of W3C Web Services Architecture

Category	Information	Rating
Specification phase	W3C Working Draft	**LOW**
Open standard	YES	**HIGH**
Potential to become open standard		**N/A**

Category	Information	Rating
Rate of advancement	Publication date: **August 2003**	**HIGH**
Potential impact on Web Services		**HIGH**
Maturity level of consortium	9 years	**HIGH**
Number of implementations		**N/A**

3.3.1.4 Recommendation

Level 2: Emerging

The main justification for this recommendation level is the immaturity of the specification. However, we believe that the work that is being done by the W3C Web Services Architecture Working Group is highly important and bears close watching. We anticipate that the direction that this Working Group is imparting will have a major impact on both the architectural direction of the World Wide Web, and the Web product landscape.

3.3.2 Global XML Web Services Architecture (GXA)

The Global XML Web Services Architecture (GXA) is an application-level protocol framework built on the foundation of XML and SOAP that helps satisfy the need for consistent support of more secure Web Services at the levels of inter-enterprise trust and business policy agreement. The GXA specifications are authored primarily by Microsoft, IBM, and Verisign, with additional authorship by organizations such as BEA Systems, RSA Security, TIBCO and SAP. GXA grew out of an April 2001 Web Services Workshop conducted by W3C whose aim was to explore the direction the W3C should take to standardize the emerging Web Services architecture.

In addition to security, GXA also covers important aspects of Web Services such as transactions, reliable messaging, and Web Services addressing. In doing so, the GXA specifications build on each other to provide the necessary functionality for a given concentration area – and they can be adopted piecemeal or en masse. The GXA specifications also leverage existing specifications such as ITU-T X.509, W3C XML Signature, and W3C XML Encryption for providing required functionality. Because GXA builds on XML and SOAP, it is "transport and transfer protocol neutral" – that is, it does not care whether the transport/transfer mechanism used for sending/receiving of SOAP messages is HTTP, SMTP, FTP, or another such mechanism.

3.3.2.1 Specification and Status

This will be listed in each section in which an individual GXA specification is described.

3.3.2.2 Main Concepts

Concentration Areas

Several of the GXA specifications will be discussed in various sections of this report. The specifications can be organized into the following concentration areas:

- **Security:** Defines a standard framework for implementing Web Services security, as well as trust and federation mechanisms.
- **Policy:** Provides general-purpose mechanisms for expressing enterprise security policies, as well as specific areas in which policies may be defined and how to associate policies with XML messages and WSDL elements.
- **Message Routing:** Describes mechanisms for dynamic routing of SOAP messages among SOAP nodes.
- **Transaction/Coordination:** Addresses multiparty Web Service interaction that requires coordination and transactional capabilities to be expressed among participating Web Services.
- **Discovery/Metadata:** Provides mechanisms for Web Services discovery and the exchange of Web Services metadata between parties.
- **Reliable Messaging:** Defines mechanisms for ensuring the reliable delivery of messages, and a framework for identifying Web Services endpoints.

The Specifications

The GXA specifications are as follows:

Table 3.2 GXA Specifications

Concentration Area	Specification	Brief Description
Security	WS-Security	Known as the "Cornerstone of GXA"; defines a standard set of SOAP extensions that implement message-level integrity and confidentiality for secure message exchanges.
	WS-Trust	Defines an extensible model for setting up and verifying trust relationships, allowing Web Services to agree on which security servers they "trust" and to rely on these servers.
	WS-SecureConversation	Leverages WS-Security and WS-Trust to provide a "context authentication model" for communications sessions.
	WS-Federation	Leverages WS-Trust and WS-SecureConversation to enable a set of organizations to establish a single, federated security domain.
Policy	WS-Policy	Provides a general-purpose specification for expressing enterprise security policies.
	WS-PolicyAssertions	Provides policies for aspects such as character encoding, natural (spoken) language, and specification versions.
	WS-SecurityPolicy	Builds on WS-Security by defining how to describe policies related to various features defined in the WS-Security specification for such aspects as security tokens, message integrity, and message age.
	WS-PolicyAttachment	Specifies how to associate a policy set with XML messages and WSDL elements (operations and portTypes).
Message Routing	WS-Routing	Describes mechanisms for routing SOAP messages without the need to rely on underlying transport mechanisms.

Concentration Area	Specification	Brief Description
	WS-Referral	Describes mechanisms by which the message paths specified in WS-Routing can be dynamically discovered.
Transaction / Coordination	WS-Coordination	Defines a general mechanism for starting and agreeing on the outcome of multiparty, multi-message Web Service tasks.
	WS-AtomicTransaction	Defines a specific set of protocols that plug into the WS-Coordination model to implement traditional two-phase atomic transaction protocols.
	WS-BusinessActivity	Defines a specific set of protocols that plug into the WS-Coordination model to implement long-running, compensation-based transaction protocols.
Discovery / Metadata	WS-Inspection	Provides a language that enables flexible discovery of Web Services, regardless of the mechanism used to describe them (WSDL, UDDI, etc.).
	WS-MetadataExchange	Provides a set of Web Service mechanisms for exchanging policies, WSDL documents, and potentially other metadata between two or more parties.
Reliable Messaging	WS-ReliableMessaging	Defines mechanisms that enable Web Services to ensure delivery of messages over unreliable communication networks.
	WS-Addressing	Provides transport-neutral mechanisms for identifying Web Service endpoints and securing end-to-end endpoint identification in messages.

3.3.2.3 Assessment

An assessment will be provided in each section in which an individual GXA specification is described.

3.3.2.4 Recommendation

Recommendations for individual GXA specifications will be provided in each section in which an individual GXA specification is described.

As a whole, we believe that GXA bears close watching because of the wide range of functionality that it covers, and also because its specifications have already begun moving into open standards consortiums (WS-Security is developing under OASIS). We believe that the functionality covered by the GXA specifications provides much-needed capabilities that are vital to the advancement of Web Services in multiple areas. We are somewhat concerned, however, about the slow pace at which the GXA specifications are being advanced into open standards consortiums, and hope that this situation improves in the near future. If the GXA specifications are not advanced at a quicker pace, they run the risk of being "replaced" by other specifications in the functionality areas that they cover.

3.3.3 Electronic Business XML (ebXML)

ebXML was an 18-month initiative that ended in May 2001 that produced a modular suite of specifications that enable enterprises of any size and in any geographical location to

conduct business over the Internet. It was sponsored by the United Nations Centre for Trade Facilitation and Electronic Business (UN/CEFACT) and OASIS. The ebXML framework provides companies with a standard method to exchange business messages, conduct trading relationships, communicate data in common terms and define and register business processes.

3.3.3.1 Specification and Status

This will be listed in each section in which an individual ebXML specification is described.

3.3.3.2 Main Concepts

ebXML "Stack"

The ebXML "stack" is comprised of five major components:

- **Business Processes:** The ebXML Business Process Schema Specification (BPSS) provides a standard framework for business process specification, and defines a mechanism for the choreography of business transactions into business collaborations.
- **Registry:** ebXML Registry provides a mechanism by which XML artifacts can be stored, maintained, and automatically discovered.
- **Messaging:** The ebXML Messaging Service (ebMS) specification specifies SOAP extensions that provide security and reliability features necessary to support international electronic business.
- **Collaboration Profiles and Agreements:** The ebXML Collaboration Protocol Profile and Agreement (CPP/A) specification provides a mechanism for defining interactions between two parties engaging in a specified business collaboration.
- **Core Components:** The Core Components Technical Specification (CCTS) provides a way to identify, capture, and maximize the reuse of business information to support and enhance interoperability across multiple business situations.

Conceptual Overview

The following figure demonstrates the conceptual overview of ebXML, and how the above components interact:

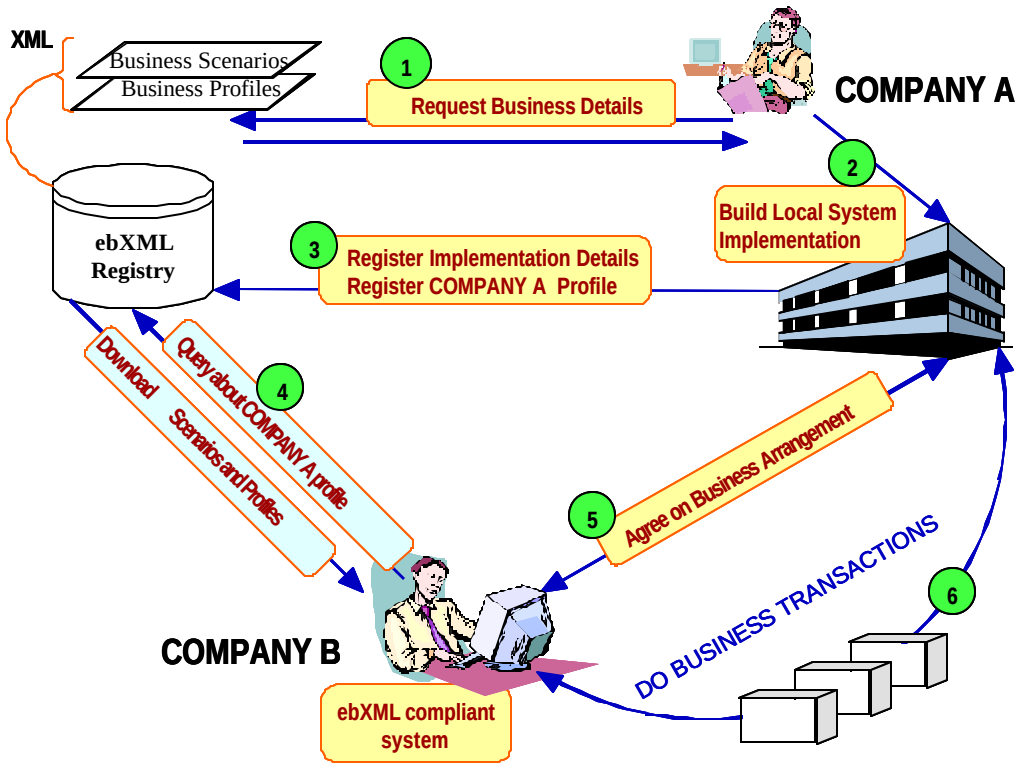

Figure 3.8 ebXML Conceptual Overview

Source: ebXML Technical Architecture v1.04

In summary, the above figure depicts a company (Company A) that submits its business profile information (step 3) to an ebXML Registry. This information is contained in a Collaboration Protocol Profile (CPP). Another company (Company B) discovers Company A's information in the ebXML Registry (step 4), and wishes to do business with Company B. Company B submits a proposed business arrangement (in the form of a Collaboration Protocol Agreement, or CPA) directly to Company A's ebXML compliant software (step 5). Company A accepts the business agreement, and Company A and B are now ready to engage in eBusiness using ebXML (step 6).

Since May 2001, further development of the ebXML specifications has been divided between OASIS and UN/CEFACT. However, recent events have caused the UN/CEFACT specifications to be moved back to OASIS where their development will continue.

ebXML and Web Services

The ebXML initiative began while Web Services was in its infancy; therefore not all aspects of ebXML are centered on Web Services. For example, the emerging OASIS Web Services Business Process Execution Language (WS BPEL) is centered on the Web Services hierarchy presented in WSDL (portType, operation, message, etc.) and networks of interacting services, while the ebXML Business Process Schema Specification (BPSS) is centered on the concept of business process patterns, workflows, and transactions between business collaboration partners. The ebXML Messaging Service (ebMS) specification, whose foundation is SOAP, is currently being adapted to support WSDL interfacing as well. Many of the ebXML specifications have incorporated Web Services since their

original version – one example is the inclusion of a SOAP interface to ebXML Registry in its 2.0 version.

In terms of adoption, we have observed that ebXML as a whole is currently most widely adopted in Europe and Asia and much less in the U.S. where it is being used in the automotive industry, healthcare, and the federal government. We believe that its low adoption here in the U.S. is partly due to a perception that ebXML and Web Services are mutually exclusive approaches.

3.3.3.3 Assessment

An assessment will be provided in each section in which an individual ebXML specification is described.

3.3.3.4 Recommendation

Recommendations for individual ebXML specifications will be provided in each section in which an individual ebXML specification is described.

As a whole, we are not certain at this time whether U.S. adoption of ebXML will increase in the future. We believe that OASIS will need to drive an effort to clear up the current confusion over the relationship between ebXML and Web Services. Since ebXML as a whole is over 2 years old, at this point it may be difficult for it to advance further as a holistic framework here in the U.S. We do not recommend that DISA adopt the ebXML framework as a whole, but instead consider individual specifications.

3.3.4 General Recommendations

As adoption of Web Services has grown, the need to define more concrete architectures has grown as well. There are multiple efforts to do so that bear close watching.

The following is a summary of the recommendations given in this section:

- We believe that the work that is being done by the W3C Web Services Architecture Working Group is highly important and bears close watching
- The Global XML Web Services Architecture (GXA) specifications also bear close watching because of the wide range of functionality that they cover, and because GXA specifications have already begun moving into open standards consortiums
- We believe that the functionality covered by the GXA specifications provides much-needed capabilities that are vital to the advancement of Web Services in multiple areas, but are somewhat concerned about the slow pace at which the GXA specifications are being advanced into open standards consortiums
- We do not recommend that DISA adopt the ebXML framework as a whole, but instead consider individual specifications

3.4 References

W3C Web Services Definition Language (WSDL) Version 1.1:
http://www.w3.org/TR/wsdl

Service-Oriented Architectures:
http://www.service-architecture.com/web-services/articles/service oriented_architecture_soa_definition.html

W3C Web Services Architecture:
http://www.w3.org/TR/2003/WD-ws-arch-20030808/

"Web Services and Peer-to-Peer Computing: Companion Technologies":
http://www.webservicesarchitect.com/content/articles/samtani05.asp

"Asynchronous Web Services and the Enterprise Service Bus":
http://www.webservices.org/index.php/article/articleview/352/1/1/

Globus Alliance:
http://www.globus.org/about/faq/general.html#globus

Joseph M. Chiusano, "Web Services Security and More: The Global XML Web Services Architecture (GXA)", Developer.com, March 2003
ebXML: www.ebxml.org

ebXML Technical Architecture Version 1.04:
http://www.ebxml.org/specs/ebTA.doc

4 Web Services and Messaging

Web Services are defined by the messages that they send or receive. This allows for loosely coupled distributed systems. There are several key aspects about messaging. Messages must be transported from one machine to another across the network. The messages need to support encoded attachments such as image or media files. Services must be able to have reliable messaging capabilities. There is a desire to reduce the bandwidth requirements by reducing the size of the messages. These aspects will be discussed in detail in this section.

4.1 Transfer Protocols

Web Services are built upon messages. The messages must be transported between machines to achieve the distributed nature of Web Services. This demands that a common transfer protocol be defined. There are several well-defined Internet transfer protocols that can be leveraged by Web Services

4.1.1 HTTP and Other Known Transfer Protocols

The Web Services Architecture indicates that services are invoked and provide results via messages that are exchanged over some communication medium. It does not specify which protocol should be used, or that only a single protocol will work. This is allows for Web Services to leverage new protocols as they are created. The current Internet standard for communication is the Hypertext Transfer Protocol (HTTP version 1.1, 1999, ftp://ftp.isi.edu/in-notes/rfc2616.txt). HTTP is based on a stateless transaction consisting of

- Connection -- The establishment of a connection by the client to the server
- Request -- The sending, by the client, of a request message to the server
- Response -- The sending, by the server, of a response to the client
- Close -- The closing of the connection by either both parties

This transaction is synchronous. The majority of Web Services are available via HTTP.

Other transfer protocols have been suggested for use by Web Services. This list includes File Transfer Protocol (FTP), Simple Mail Transfer Protocol (SMTP) and Blocks Extensible Exchange Protocol (BEEP.) FTP and SMTP are extensively used to perform the tasks they were designed to do. There are issues with each protocol that makes them poorly suited for Web Services. This does not technically prevent them from being used for Web Services, but in reality, no one is using them for Web Services. BEEP is a new specification that looks promising but is not gaining any ground on HTTP.

Although HTTP is the transfer protocol for Web Services, it does have its limitations. These limitations have led to vendors developing proprietary or now standard protocols. The main deficiencies of HTTP are its synchronous nature, its lack reliable messaging and the fact that it is stateless.

4.1.2 Message-Oriented Middleware (MOM)

Message-Oriented Middleware (MOM) (typically implemented as message queues) is a mature technology that can enable message flow between systems within an enterprise. MOM architectures are designed to work behind the firewall. They provide reliability, scalability, performance, asynchronous capability and transaction support. There are many vendors that provide MOM implementations such as Sonic's SonicMQ, IBM's MQSeries or Microsoft's MSMQ.

The downside to using a MOM's is that all clients of the system must be leveraging the same implementation because each one uses its own proprietary transfer protocol. Since the MOM architecture is designed to run within a single enterprise this typically is not an issue. For Java developers the Java Message Service (JMS) provides a layer of abstraction making it easier to change MOM implementations. SOAP messages can be delivered with the added benefits that MOM provides. For enterprise wide solutions, MOM can be the transfer mechanism.

There is a symbiotic relationship between HTTP-base Web Services and MOM. The HTTP protocol exposes the services beyond the enterprise passing through the firewall. The MOM provides backbone architecture allowing enterprise clients full access to the MOM capabilities. The HTTP Web Service can be implemented as a client to the MOM system, basically exposing the enterprise services as Web Services. This allows for a greater variety of clients to access the services.

4.1.3 Recommendation

HTTP is the defined transfer protocol standard for Web Services because of the ubiquity of HTTP implementations in today's web servers. An HTTP Web Service can be used in conjunction with a MOM implementation to provide the benefits of both to a wide range of clients, both inside and outside the enterprise. The benefits of MOM to support reliable messaging cannot be understated, however, and standards to support interoperable reliable messaging are likely to appear. Reliable messaging standards efforts are discussed later in this section.

4.2 Messaging and Attachments

A SOAP message may need to be transmitted with attachments of different kinds ranging from engineering drawings to audio files. For example a meteorological Web Service could provide current precipitation as an image along with the XML document containing forecast information. The Amazon.com Web Service allows clients to retrieve thumbnail images of books available for sale. There have been various ideas on how messages should be associated with various formatted attachments.

4.2.1 Specification and Status

SOAP with Attachments (SwA) W3C Note, December 2000

http://www.w3.org/TR/SOAP-attachments

4.2.2 Main Concepts

The two main specifications for handling message and attachments were SOAP with Attachments (SwA) and WS-Attachments. SwA is a W3C Note that defines how a SOAP 1.1 message can be carried inside a MIME multipart/related message while still allowing the processing rules for the message to be preserved. It leverages the MIME multipart mechanism to provide a means of attaching additional parts to the message. It does this without having to modify or introduce additional specifications. The WS-Attachments specification leverages Direct Internet Message Encapsulation (DIME), which is being abandoned in favor of MIME multipart based messages. While there are still some vendors supporting WS-Attachments and DIME the majority of implantations are using SwA.

SwA works within the SOAP and MIME standards to define how SOAP messages can be associated with one or more attachments that are transported in their native format. Most Internet communication protocols already transport MIME encoded content. Attachments are referenced using href attributes defined in SOAP 1.1. Resolution of URIs, including href attributes, in SOAP messages are done using the well-defined rules for multipart MIME messages.

While SwA is the defined standard for dealing with attachments there is a proposal that is trying to improve upon it. The proposal is the Infoset Addendum to SOAP Messages with Attachments. The proposal builds on SwA and is backwards compatible which is a major advantage. It aligns the XML Infoset-based data model and the SOAP processing model. A key concept is an additional XML Schema complexType that extends the xs:base64Binary type with an xmime:mediaType attribute. This allows for MIME type to be defined for the base64-encoded content. This allows for the attachments to be embedded in the XML body rather than existing outside of the message with a URI reference to them. The proposal still supports web references to external content but they are not required. This proposal is trying to address the desire to integrate pre-existing data formats within a XML document.

4.2.3 Recommendation

Level 1: Ready for use.

SOAP with Attachments is the standard way of handling attachments with SOAP messages. There are a large number of implementations behind this open speciation including SOAP with Attachments API for Java (SAAJ) and Systinet WASP Server for C++.

4.3 Reliable Messaging

Reliable messaging is not a new problem that was created by Web Services. Shipping companies provide tracking numbers so that you can know that your package was delivered and signed for. Instead of shipping packages, Web Services sending millions of messages in the span of a few minutes. It is critical to business and war fighters alike that messages be reliably delivered to their destinations.

4.3.1 Specification and Status

OASIS Message Service Specification Version (ebMS) 2.0, April 2002.
http://www.oasis-open.org/committees/ebxml-msg/documents/ebMS_v2_0.pdf

OASIS Web Services Reliability Services (WS-Reliability) Working Draft 0.52, September 2003.
http://www.oasis-open.org/committees/download.php/3549/WS-Reliability-2003-09-05b.pdf

Web Services Reliable Messaging Protocol (WS-ReliableMessaging), March 2003.
http://msdn.microsoft.com/ws/2003/03/ws-reliablemessaging

Web Services Acknowledgement (WS-Acknowledgement) 0.91, February 2003.
http://dev2dev.bea.com/technologies/webservices/WS-Acknowledgement_Intro.jsp

4.3.2 Main Concepts

At the simplest level, reliable messaging can be defined as the ability of a sender to deliver a message once and only once to a receiver. This definition can be expanded to several requirements:

- Leverage SOAP to define the reliability mechanism
- Messages must not be lost if systems go offline
- Assert that messages can be received at least once, at most once, exactly once, etc
- Messages will be delivered in the order they are sent
- Sender and receiver will be notified when a message was not delivered
- Allow multiple hops between the sender and receiver
- Be independent from the transport protocol

Since SOAP is the standard for Web Services messaging, the reliability protocol should be compatible with SOAP. SOAP's extensibility mechanism should be leveraged. By using do so, reliability will be encapsulated in the SOAP message and will ensure that it is independent of the transport protocol. It is extremely likely that in the life time of a Web Service that there will be a system failure or system maintenance which will cause downtime. Message reliability dictates that the messages must not be lost. The message should be resent a specified number of times or for a period time. If at the end the message has not been received, the sender must receive an indication that the message was not delivered. Messages that were being processed by the service that went down must be persisted, so when it is back on line, it can continue processing. The ability to indicate that messages can be received an exact number of times and arrive in the correct order requires unique message identification and sequence identification. Web Services do not define the transport protocol and even allow multiple protocols to be used to deliver a single message. Hence the reliable messaging solution must be independent from the transfer protocols.

There are current solutions that provide reliable messaging. Reliability can be implemented at the service level, which places the burden on every service. The problem with both situations is that the solution is not interoperable with other systems or services. There are multiple specifications that are attempting to define the reliable messaging standard. The list of specifications includes ebMS, WS-Reliability, WS-

ReliableMessaging and WS-Acknowledgement. While the WS-Reliability is at the OASIS Web Services Reliable Messaging TC, it is only a working draft. The differences between the specifications are minor. There is no technical advantage to distinguish between the specifications.

4.3.3 Recommendation

Level 2: Emerging.

Reliable messaging is a known necessity for Web Services. At this time there is no standard for implementing reliable messaging. There are a number of venders who can currently provide systems that offer reliable messaging but there is no market leader and the implementation will not be interoperable with other systems. This area should be closely watched to see what standard emerges from the possible choices.

4.3.4 WS-Routing/WS-Referral (GXA)

The WS-Routing specification is part of the Global XML Web Services Architecture (GXA). It was authored by Microsoft. They are discussed together here because of their close dependencies. WS-Routing specifies mechanisms by which message routing details can be specified in SOAP messages, while WS-Referral specifies mechanisms by which the message paths specified in WS-Routing can be dynamically discovered.

4.3.4.1 Specification and Status

This section references the following specifications:

- WS-Routing Version 1.0 (October 2001)
- WS-Referral Version 1.0 (October 2001)

4.3.4.2 Main Concepts

SOAP Intermediaries/Routers

SOAP by itself does not define an actual message path along which a SOAP message is to travel – rather, it relies on its underlying application layer protocols (such as HTTP or SMTP) to do so. However, these protocols each have their own mechanisms for defining message paths. As GXA is transport-neutral, it cannot rely on SOAP's underlying application layer protocols for specifying message paths. The WS-Routing specification leverages the concept of a "SOAP intermediary" as defined in SOAP 1.2, therefore "raising up" the capability to define a message path to the SOAP layer and making it transport-neutral.

WS-Routing and WS-Referral define the concept of a *SOAP router*, which is a SOAP node that exposes SOAP message relaying as a Web Service either as a standalone service or in combination with other services. WS-Routing also defines a new SOAP header named <path> that contains the following elements:

- **from:** Denotes the message originator
- **to:** Denotes the ultimate receiver
- **fwd:** Contains the forward message path
- **rev:** Contains the reverse message path, enabling 2-way message exchange

The <wsrp:fwd> and <wsrp:rev> elements also contain <wsrp:via> elements to denote intermediaries.

Example SOAP Header

The following example demonstrates a "path" SOAP header:

```
[01] <SOAP-ENV:Header>
[02]   <wsrp:path>
[03]     <wsrp:to>soap://D.com</wsrp:to>
[04]       <wsrp:fwd>
[05]         <wsrp:via>soap://B.com</wsrp:via>
[06]         <wsrp:via>soap://C.com</wsrp:via>
[07]       </wsrp:fwd>
[08]     <wsrp:from>soap://A.com</wsrp:from>
[09]     <wsrp:id>uuid:84b9f5d0-33fb-4a81-b02b-5b760641c1d6</wsrp:id>
[10]   </wsrp:path>
[11] </SOAP-ENV:Header>
```

In the above example, the message is to travel from `soap://A.com` [line 08] to `soap://D.com` [line 03], passing through ("via") intermediaries `soap://B.com` [line 05] and `soap://C.com` [line 06]. This may be done for purposes such as load balancing, if a Web Service's network address has changed, or for dynamic message path optimization (a "better" path suddenly exists).

Referral Statements

The basic unit of WS-Referral is a *referral statement*, which is an XML-based structure for describing a routing entry along with a set of conditions under which the statement is satisfied. A referral statement contains five parts – among them are:

- A set of SOAP actors for which a statement is intended
- A set of conditions that have to be met for a statement to be satisfied
- A set of SOAP routers that a statement is referring to as part of the delegation

Example Referral Statement

The following example illustrates a referral statement:

```
[01] <r:ref>
[02]   <r:for>
[03]         <r:prefix>soap://a.org</r:prefix>
[04]   </r:for>
[05]   <r:if>
[06]     <r:ttl>43200000</r:ttl>
[07]   </r:if>
[08]   <r:go>
[09]     <r:via>soap://b.org</r:via>
[10]   </r:go>
[11]   <r:refId>uuid:09233523-345b-4351-b623-5dsf35sgs5d6</r:refId>
[12] </r:ref>
```

In the above example, the <r:prefix> element [line 03] means that any SOAP node beginning with the contained URI is considered to be a match for the referral. The <r:ttl> element [line 06] denotes "time to live" – that is, it sets a "time to live" limit on the availability of a referral. The value of this element (43200000 [line 06]) is the number of mil-

liseconds, which equates to 12 hours. Therefore, the entire statement would be read as follows: "for all SOAP nodes that begin with the URI soap://a.org, if this referral is less than 12 hours old, then go to soap://b.org [line 09]".

4.3.4.3 Assessment

Table 4.1 Assessment of WS-Routing/WS-Referral

Category	Information	Rating
Specification phase	Initial public draft release	**LOW**
Open standard	NO	**LOW**
Potential to become open standard		**MEDIUM**
Rate of advancement	• WS-Routing publication date: **October 2001** • WS-Referral publication date: **October 2001**	**LOW**
Potential impact on Web Services		**HIGH**
Maturity level of consortium		**N/A**
Number of implementations	None	**LOW**

4.3.4.4 Implementations

None.

4.3.4.5 Recommendation

Level 3: Questionable

Although we believe that WS-Routing and WS-Referral can have a high impact on Web Services, they are not being developed within an open standards consortium. Additionally, the rate of advancement of each of these specifications is low, and it is unclear at this time whether Microsoft will advance them further.

4.3.5 Binary XML and XML Compression

XML is one of the building blocks for Web Services. It is well defined and widely adopted in the industry. There are desires to either have a binary format or to use a compression library to reduce the bandwidth used by Web Services.

4.3.5.1 Specification and Status

There are no standards currently defined for Binary XML or XML Compression.

4.3.5.2 Main Concepts

There is a need for smaller message sizes in areas that have restricted bandwidth. Smaller sizes will also increase throughput for services that require high message vol-

ume. There are two ways to achieve small message sizes, use a binary format for XML or compress the XML message.

There has been interest in a binary format of XML for a long time. A native binary encoding solution avoids the additional conversion from text to binary. This conversion consumes additional processing power and memory. A binary XML standard would allow binary encoding to be the natural language that parsers dealt with. A standard would allow off the shelf components to generate binary structures that could be used for storage and transport. The W3C Workshop on Binary Interchange of XML Information Item Sets was held at the end of September 2003. This represents progress towards a standard, but is only the first steps.

The second way to decrease the message size is to use compression. The nature of XML leads high levels of compression. The downside to compression is that there is an increase in processing time to compress and decompress the messages on each side. Some of the standard compression tools such as gzip or bzip2 can be used. The problem with these is that they are not designed for XML so they cannot take advantage of the nature of XML. Domain specific compression algorithms such as the ones used by the Web3D Consortium's Extensible 3D (X3D) have achieved compression factors upwards of 30 to 1. This is much greater than what gzip can do. A generic XML compression scheme should be able to leverage XML Schema to obtain high levels of compression. Unfortunately there is no standard for how this should be done, or how to define that the Web Service is using compressed messages.

4.3.5.3 Recommendation

Level 3: Questionable

There is no standard for binary XML or compressing the messages. There are products that will generate proprietary binary XML messages and/or compression algorithms including XMill, gzip or bzip2 that will compress the XML message before it is sent. Any of the products or tools will work, but they will lead to interoperability problems.

5 Web Services and Security

This section focuses on Web Services and security. When Web Services are used within an organization (i.e. behind the firewall), security is not a great issue. However, once Web Services-based exchanges branch out beyond an organization's firewall and span across organizations, security becomes a very large factor. Over the past several years, various specifications have begun to emerge to address the issue of Web Services and security. We address these specifications in this section.

We begin with a discussion of various *security functionality categories*, which are high-level security areas that address various aspects of digital security. We believe it is necessary that open standards provide mechanisms to address the various issues associated with each of these categories as applied to Web Services. We then discuss emerging and approved specifications in each security functionality category.

The following table lists the security functionality categories covered in this section:

Table 5.1 Security Functionality Categories

Category	Description
Authentication	Allowing individual users and organizations to validate the identity of each party in a Web-based transaction.
Identity Management	Involves the management of electronic user identities across multiple applications.
Integrity	Ensuring that a message or document that is digitally signed has not been changed or corrupted in transit.
Confidentiality	Protecting information from interception during transmission.
Authorization	Controlling access privileges to resources.
Non-repudiation	The ability to ensure that a party to a contract or communication cannot deny the authenticity of their signature on a document or the sending of a message that they originated.
Trust	The characteristic that one entity is willing to rely upon a second entity to execute a set of actions and/or to make set of assertions about a set of subjects and/or scopes.
Policy	A set of information that conveys various characteristics of, and rules for, a Web Service, such as authorization and the types of security tokens that a Web Service accepts.

We begin this section with a description of an emerging OASIS standard that provides a framework for Web Services security through its specification of a standard mechanism by which various security standards can be incorporated into a SOAP header. This emerging OASIS standard is called *Web Services Security (WS-Security)*.

5.1 Security Framework - OASIS Web Services Security (WS-Security)

The WS-Security specification is part of the Global XML Web Services Architecture (GXA). It was originally released in October 2001 and authored by Microsoft, IBM, and Verisign. WS-Security was submitted to OASIS in June 2002 where it is now being developed under the OASIS Web Services Security TC. WS-Security forms the basis for many other GXA specifications, and thus is considered "The Cornerstone of GXA."

Many Web Service interactions today utilize *Secure Socket Layer/Transport Layer Security (SSL/TLS)* for their transmission security requirements. While this technique works well in point-to-point scenarios, there is a need to maintain secure contexts over multi-point message paths where trust domains need to be crossed (such as between organizations)—that is, to support end-to-end message-level security, not just transport-level security. WS-Security addresses this need.

5.1.1 Specification and Status

This section references the following specifications:

- WS-Security: SOAP Message Security (OASIS TC Approved Specification, August 2003)
- WS-Security: Username Token Profile (OASIS TC Approved Specification, August 2003)
- WS-Security: X.509 Token Profile (OASIS TC Approved Specification, August 2003)

These three specifications are in OASIS public review until 19 October 2003, at which time they may become OASIS standards.

5.1.2 Main Concepts

Secure Message Exchanges

WS-Security defines a standard set of SOAP extensions that implement message-level integrity and confidentiality for secure message exchanges. It also provides a general-purpose mechanism for associating *security tokens* with message content, and it supports multiple security token formats. WS-Security is designed to support a wide variety of security models—i.e. it is designed to support multiple security token formats, multiple trust domains, multiple signature formats, and multiple encryption technologies. This includes existing security models, as well as security models that may be released in the future. WS-Security supports several of the security functionality categories discussed above, including authentication, integrity, confidentiality, and non-repudiation.

The following figure demonstrates how WS-Security enables the maintenance of a secure context over a multi-point message path. It denotes three Web participants—a "sender" Web Service, an "intermediary" Web Service, and a "receiver" Web Service. Rather than carrying a separate security context from one participant to another (as would be necessary using SSL/TLS), WS-Security allows for the security context to be carried over the entire interaction as a *"security umbrella":*

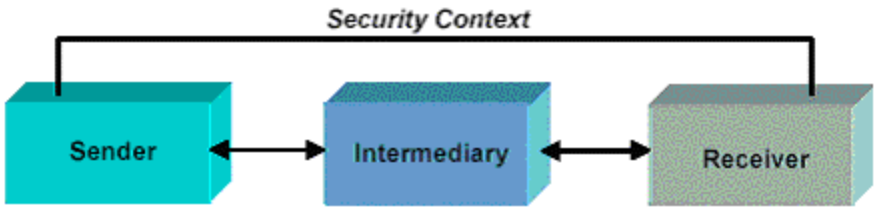

Figure 5.1 WS-Security Context Participants

Source: Joseph M. Chiusano, "Web Services Security and More: The Global XML Web Services Architecture (GXA)", Developer.com

Security Tokens

Examples of security tokens that WS-Security supports are an X.509 certificate or a username/password.

The "X.509 Token Profile" specification describes the use of the X.509 authentication framework with the SOAP Message Security specification (the "core" specification). An X.509 certificate may be used to validate a public key that may be used to authenticate a WS-Security-enhanced message or to identify the public key with which a WS-Security-enhanced message has been encrypted.

The following example illustrates the specification of an X.509 certificate using WS-Security:

```
[01]  <?xml version="1.0" encoding="utf-8"?>
[02]  <S:Envelope
[03]     <S:Header>
[04]         <wsse:Security>
[05]             <wsse:BinarySecurityToken
[06]                 wsu:Id="X509Token">
[07]                 ValueType="wsse:X509v3"
[08]                 EncodingType="wsse:Base64Binary"
[09]                 MIIEZzCCA9CgAwIBAgIQEmtJZc0rqrKh5i...
[10]             </wsse:BinarySecurityToken>
[11]              <ds:Signature>
[12]                      .....
[13]                  <ds:SignedInfo>
[14]                     <ds:Reference URI="#X509Token">…</ds:Reference>
[15]                 </ds:SignedInfo>
[16]                 <ds:SignatureValue>BL8jdfToEb1l/vXcMZNNjPOV...
[17]             </ds:SignatureValue>
[18]                    .....
[19]             </ds:Signature>
[20]         </wsse:Security>
[21]     </S:Header>
[22]     <S:Body wsu:Id="MsgBody">  .....  </S:Body>
[23]  </S:Envelope>
```

In the above example, the X.509 certificate information is specified within a `<wsse:BinarySecurityToken>` element [lines 05-10]. The certificate is referenced by URI within a signature, using W3C XML Signature [line 14]. The `<wsse:Security>`

element [line 04] is the "main" WS-Security element into which all WS-Security SOAP extensions are placed.

The "Username Token Profile" specification describes how to use a username token (and optionally a password) with the SOAP Message Security specification.

The following example illustrates the specification of a username and password using WS-Security:

```
[01] <?xml version="1.0" encoding="utf-8"?>
[02] <S:Envelope
[03]     <S:Header>
[04]         <wsse:Security>
[05]             <wsse:UsernameToken>
[06]                 <wsse:Username>Zoe</wsse:Username>
[07]                 <wsse:Password>MyPassword</wsse:Password>
[08]             </wsse:UsernameToken>
[09]                .....
[10]         </wsse:Security>
[11]     </S:Header>
[12]       .....
[13] </S:Envelope>
```

In the above example, the username and password are specified within a `<wsse:UsernameToken>` element [lines 05-08]. Because the password in the above example is clear text [line 07], it should be sent on a secured channel; otherwise, it should be obscured by creating a password digest.

Upcoming Features

Now that the above specifications have been completed, the OASIS WS-Security TC has begun focusing on the creation of various token profile specifications, including Kerberos, XrML and SAML.

Message Integrity and Confidentiality

WS-Security also provides message integrity and confidentiality through its use of the W3C XML Signature and W3C XML Encryption specifications. Message integrity is provided by XML Signature in conjunction with security tokens to ensure that modifications to messages are detected, while message confidentiality leverages XML Encryption in conjunction with security tokens to keep portions of a SOAP message confidential.

More is Required Beyond Message-Level Authentication

It should be noted that while the message-level authentication mechanisms provided by WS-Security are essential, more is required than just message-level authentication. That is, the other aspects of security discussed in this document (session-level authentication, access control mechanisms, security policies, etc.) – along with message-level authentication - serve to provide a comprehensive security solution.

5.1.3 Assessment

Table 5.2 Assessment of WS-Security

Category	Information	Rating
Specification phase	OASIS TC Approved Specification	**MEDIUM**

Category	Information	Rating
Open standard	YES	**HIGH**
Potential to become open standard		**N/A**
Rate of advancement		**N/A**
Potential impact on Web Services		**HIGH**
Maturity level of consortium	10 years	**HIGH**
Number of implementations	1	**LOW**

5.1.4 Implementations

- Microsoft Web Services Enhancements (WSE): http://msdn.microsoft.com/webservices/building/wse/default.aspx

5.1.5 Recommendation

Level 2: Emerging

We believe that WS-Security will have the largest single impact on the advancement of Web Services of any of the specifications discussed in this section. The main reasons for this recommendation level are the current status of the WS-Security specifications (not yet OASIS standards), as well as a low number of implementations. If WS-Security becomes an OASIS standard and more implementations emerge, it may then be considered as "Level 1: Suitable For Use".

5.2 Authentication/Identity Management

Authentication involves allowing individual users and organizations to validate the identity of each party in a Web-based transaction. *Identity management* involves the management of electronic user identities across multiple applications. Because these security functional categories are closely related, we discuss them in the same section.

The following specifications are discussed in this section:

- OASIS Security Assertion Markup Language (SAML)
- The Liberty Alliance
- Web Services Federation Language (WS-Federation)
- OASIS XML Common Biometric Format (XCBF)

5.2.1 OASIS Security Assertion Markup Language (SAML)

OASIS Security Assertion Markup Language (SAML) is an XML-based framework for exchanging security information. It is being developed by the OASIS Security Services Technical Committee (SSTC). One major advantage of using SAML is that it allows security domains to be crossed more easily. The security information that SAML addresses is expressed in the form of *assertions* about *subjects*, where an assertion is a declaration of certain facts, and a subject is an entity (either human or computer) that has an identity in some security domain. For example, an assertion can be made that a particular client

was granted “update” privileges to a specific database resource at a certain time. This information may result in the client being granted the same privileges at a later time without being re-authenticated.

5.2.1.1 Specification and Status

This section references the following specification:

- SAML Version 1.1 (OASIS Standard, September 2003)

5.2.1.2 Main Concepts

SAML Statements

SAML uses three types of “statements” about subjects:

Authentication statement: Expresses that the issuing authority authenticated a specific subject at a given time. For example, “Subject A has been authenticated by means B at time C”.

Attribute statement: Describes specific attributes of a subject. For example, “Subject A is associated with the department Human Resources”.

Authorization decision statement: Expresses whether a given subject has been granted specific permissions to access a particular resource. For example, “Subject A has been granted permission to access resource B with privilege C at time D”.

The following example illustrates an authentication statement:

```
[01] <saml:assertion Issuer=“issuer1.com”…>
[02]     <saml:Conditions NotBefore=… NotAfter=…/>
[03]     <saml:AuthenticationStatement
[04]         AuthenticationMethod=”urn:oasis:names:tc:SAML:1.0:am:X509-PKI”
[05]          AuthenticationInstant=”2003-04-11T21:41:00Z”>
[06]       <saml:subject …>John Smith</saml:subject>
[07]     </saml:AuthenticationStatement>
[08]     <saml:AttributeStatement>
[09]         <saml:subject …>John Smith</saml:subject>
[10]         <saml:Attribute AttributeName=“Department” …>
[11]             <saml:AttributeValue>Human Resources</AttributeValue>
[12]         </saml:Attribute>
[13]         <saml:Attribute AttributeName=“ReportsTo” …>
[14]             <saml:AttributeValue>Mary Jones</AttributeValue>
[15]         </saml:Attribute>
[16]      </saml:AttributeStatement>
[17] </saml:Assertion>
```

The above example expresses that a subject named “John Smith” [line06] was authenticated by use of an X.509 public key [line 04] at a given date and time [line 05]. An attribute statement [lines 08-16] is also included that expresses several attributes of this subject, specifically that the subject works for the Human Resources department [lines 10-12] and report to Mary Jones [lines 13-15].

The following example illustrates an authorization decision statement:

```
[01] <saml:Assertion …>
[02]     <saml:Conditions …/>
[03]     <saml:AuthorizationDecisionStatement
[04]         Decision=“Permit”
```

```
[05]         Resource="http://example.com/report1.htm">
[06]         <saml:Action>read</saml:Action>
[07]         <saml:Subject>
[08]             <saml:NameIdentifier
[09]                 SecurityDomain="somedomain.com"
[10]                 Name="John Smith" />
[11]         </saml:Subject>
[12]     </saml:AuthorizationDecisionStatement>
[13] </saml:Assertion>
```

The above example expresses that a subject named "John Smith" [line 10] in security domain "somedomain.com" [line 09] has been granted "read" access [line 06] to a report found at URL "http://example.com/report1.htm" [line 05].

SAML Domain Model

The SAML Domain Model describes the mechanisms by which clients can request assertions from "SAML authorities" and receive a response from them. This model is exhibited in the following figure:

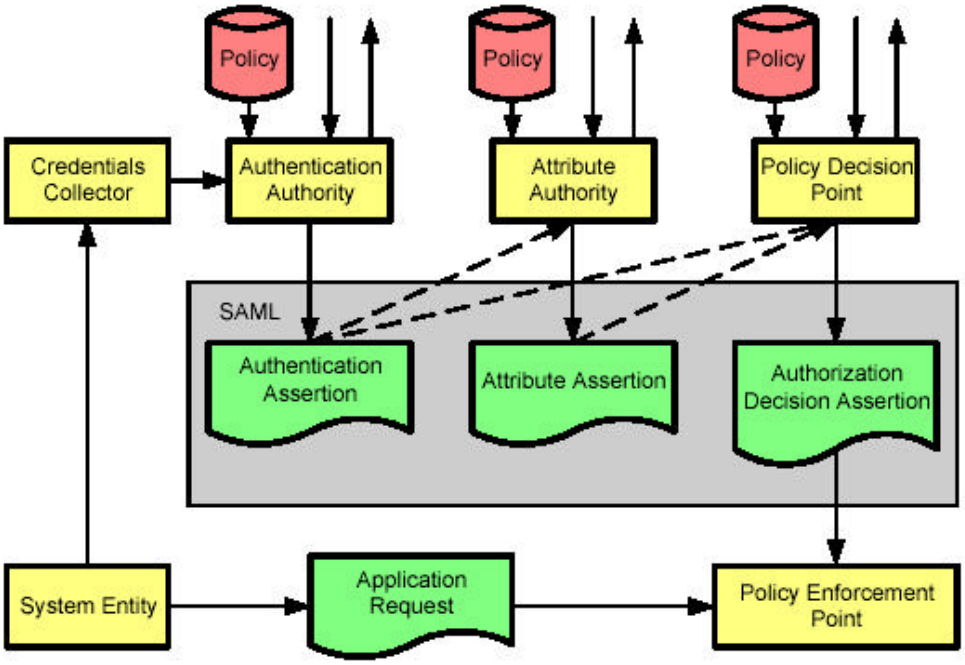

Figure 5.2 The SAML Domain Model

Source: SAML Version 1.1 Specification

The above figure illustrates three types of SAML authorities at the top:

- An *Authentication Authority* (source of authentication assertions)
- An *Attribute Authority* (source of attribute assertions)
- A *Policy Decision Point*, or *PDP* (creates authorization decision assertions based on multiple inputs)

The *Credentials Collector* requests an authentication assertion for a particular subject from the Authentication Authority. This request returns one or more statements about authentication acts that have occurred in previous interactions between the indicated subject

and the Authentication Authority. The Authentication Authority also requests one or more attribute assertions for the subject from the Attribute Authority. Additionally, the Authentication Authority queries the Policy Decision Point (PDP) providing it with the subject's attributes and the requested action by the subject, and receives an authorization decision assertion as to whether the requested action by the subject on the given resource should be allowed. A *Policy Enforcement Point (PEP)* as part of access control then enforces this authorization decision whenever an application request is made in the future regarding this subject, resource, and action.

Single Sign-On (SSO)

One major design goal of SAML is *single sign-on (SSO)* – the sharing of authentication information across different applications so that a user can authenticate in one domain and use resources in other domains without re-authenticating. This concept is shown in the following figure:

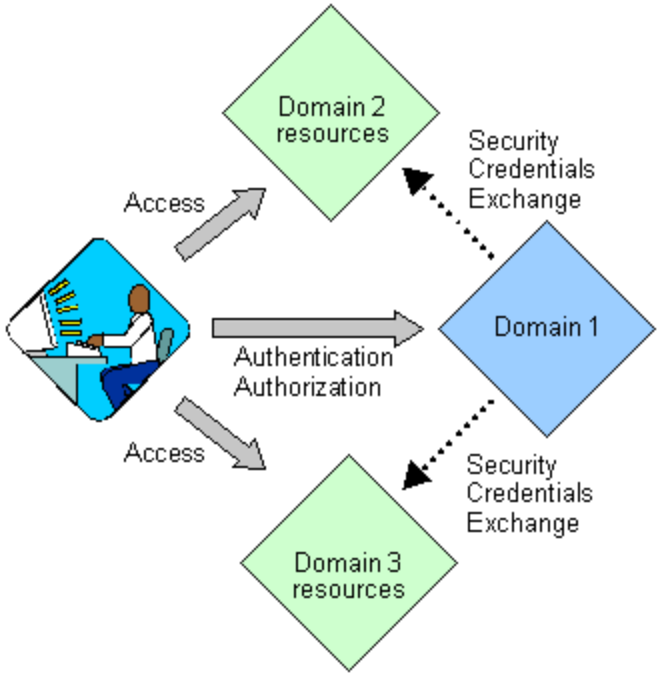

Figure 5.3 Concept of Single Sign-on

Source: Seshardri Gokul, "Authenticating Web Services with SAML", informIT.com

In the above figure, a subject is authenticated in a domain (Domain 1) and their credentials are then shared with other domains (Domain 2 and Domain 3). This alleviates the need for the subject to sign on within domains 2 and 3.

SAML and Web Services

The following two aspects of SAML and Web Services will be discussed in this section:

- SAML SOAP Binding: Defines how to use SOAP to send and receive SAML requests and responses.

- Web Services-Based Exchanges: Using SAML to secure Web Services-based exchanges.

SAML SOAP Binding

SAML currently defines only one binding: SOAP over HTTP. The system model used for SAML conversations over SOAP is a simple request-response model in which a system entity sends a SAML request to a SAML responder, which sends back a SAML response.

The following is an example of a SOAP-over-HTTP request:

```
[01] POST /SamlService HTTP/1.1
[02] Host: www.example.com
[03] Content-Type: text/xml
[04] Content-Length: nnn
[05] SOAPAction: http://www.oasis-open.org/committees/security
[06] <SOAP-ENV:Envelope
[07]     xmlns:SOAP-ENV=”http://schemas.xmlsoap.org/soap/envelope/”>
[08]     <SOAP-ENV:Body>
[09]         <samlp:Request xmlns:samlp:=”…” xmlns:saml=”…” xmlns:ds=”…”>
[10]             <ds:Signature> … </ds:Signature>
[11]             <samlp:AuthenticationQuery>
[12]                  ....
[13]             </samlp:AuthenticationQuery>
[14]         </samlp:Request>
[15]     </SOAP-ENV:Body>
[16] </SOAP-ENV:Envelope>
```

In the above example, a request [lines 09-14] is made for an assertion containing an authentication statement from a SAML authentication authority. The `<samlp:AuthenticationQuery>` element [lines 11-13] would contain the details pertinent to the assertion request.

Web Services-Based Exchanges

SAML can also be used to secure Web Services-based exchanges by authenticating requestors to Web Services, and Web Services to other Web Services. The single sign-on capabilities of SAML can also be used to authenticate a requestor across multiple Web Services. The OASIS WS-Security TC is in the process of completing a SAML profile that describes the carrying of SAML assertions in a WS-Security header.

Future Enhancements

Plans for the SAML Version 2.0 specification currently include the following:

- Incorporation of new features such as session support, exchange of metadata to ensure more interoperable interactions, and collection of credentials
- Support for full identity federation through integration of the specifications contributed to the SSTC by the Liberty Alliance

5.2.1.3 Assessment

Table 5.3 Assessment of SAML

Category	Information	Rating
Specification phase	OASIS Standard	**HIGH**

Category	Information	Rating
Open standard	YES	**HIGH**
Potential to become open standard		**N/A**
Rate of advancement		**N/A**
Potential impact on Web Services		**HIGH**
Maturity level of consortium	10 years	**HIGH**
Number of implementations	9	**MEDIUM**

5.2.1.4 Implementations

The following are three examples of identified implementations:

- SunONE Identity Server: http://xml.coverpages.org/ni2003-01-16-a.html
- Netegrity SAML Affiliate Agent: http://xml.coverpages.org/NetegrityAAgent.html
- Novell iChain: http://xml.coverpages.org/Novell-iChain.html

5.2.1.5 Recommendation

Level 1: Suitable For Use

SAML has gained wide acceptance in many different industries and settings. We attribute this wide acceptance mostly to the high impact of its capabilities not only on Web Services security, but also on digital security in general. We foresee the number of SAML implementations growing steadily in the medium- and long-term future.

5.2.2 Liberty Alliance

The Liberty Alliance Project is an initiative comprised of 160 organizations whose mission is to drive a new level of trust, commerce, and communications on the Internet through the creation of open technical specifications. Its members include organizations such as American Express, Hewlett Packard, RSA Security, Sun Microsystems, and America Online. Its federal membership includes the U.S. Department of Defense and the U.S. General Services Administration. The Liberty architecture consists of a multi-level layered specification set, based on open standards including SAML and SOAP. The vision of the Liberty Alliance is to enable a networked world in which individuals and businesses can more easily conduct transactions while protecting the privacy and security of vital identity information.

5.2.2.1 Specification and Status

This section references the following specifications:

- Liberty Alliance Architecture Overview Version 1.1 (Final, January 2003)
- Liberty Identity Web Services Framework (ID-WSF) Overview Version 1.2-06 (Draft, July 2003)

5.2.2.2 Main Concepts

Federated Network Identity/Single Sign-On

The Liberty Alliance specifications cover two main concepts:

- Federated network identity
- Single sign-on

The term network identity refers to the global set of attributes that are contained in an individual's various accounts with different service providers. It can include information such as name, phone number, Social Security Number, credit records, etc. The notion of a federated network identity means that information particular to an individual may be administered by the user and securely shared with entities of the user's choosing.

A *federated network identity model* ensures that critical private information is used by appropriate parties within a *circle of trust*. A circle of trust is a federation of service providers and identity providers that have business relationships based on Liberty architecture and operational agreements with whom users can transact business in a secure and apparently seamless environment. Circles of trust can be enterprise circles of trust or consumer circles of trust, as shown in the following figure:

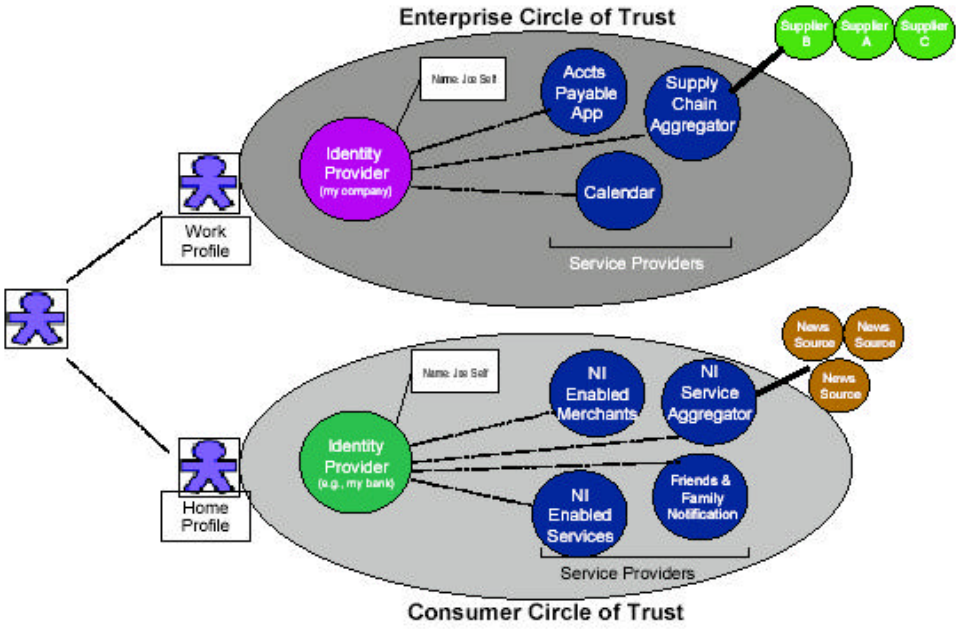

Figure 5.4

Figure 5.5 Federated Network Identity and Circles of Trust

Source: Liberty Alliance Architecture Overview Version 1.1 Specification

The combination of federated network identity and single sign-on is known as *federated single sign-on*. This capability enables users to sign-on with one member of an affiliate group and subsequently use other sites within the group without having to sign-on again.

The Liberty Alliance and Web Services

One of the major components of the Liberty architecture is the Liberty Identity Web Services Framework (ID-WSF). This component provides a basic framework for identity services, such as Web-based identity service discovery and invocation. It also provides a SOAP binding for the framework.

An *identity service* is an abstract notion of a Web Service that acts upon some resource to retrieve information about an identity, update information about an identity, or perform some action for the benefit of some identity. An example of a resource is a calendar containing appointments for a particular identity. A *discovery service* is an identity service that allows requesters to discover resource offerings – thus, it is essentially a Web Service interface for "discovery resources", each of which can be viewed as a registry of resource offerings. Entities can place resource offerings in a discovery resource, and this will allow other entities to discover these resource offerings. The discovery service can also function as a security token service, issuing security tokens to the requester that the requester will use in the request to the discovered identity service.

Once a service has been discovered and sufficient authorization data has been received from a trusted authority, the invoking entity (Web Services Consumer) may invoke the service at the hosting/relying entity (Web Services Provider).

The Liberty Alliance and SAML

The Liberty Alliance incorporated SAML into its Version 1.1 specifications introduced in 2002. The Liberty Alliance chose to extend SAML in Version 1.1 to include security enhancements vital to identity management, such as:

- Opt-in account linking
- Simple session management
- Global logout capabilities

The Liberty Alliance also contributed its Version 1.1 federated network identity specifications to the OASIS Security Services TC in April 2003, for possible incorporation of the Version 1.1 specification features (such as those described above) in future versions of SAML.

While the work of SAML and that of the Liberty Alliance may appear to conflict, it is actually complementary. Both specifications address mechanisms for single sign-on; however, SAML concentrates more heavily on the definition and handling of security assertions, while the Liberty Alliance concentrates more heavily on the notion of federated network identities. However, the planned incorporation of full identity federation into SAML Version 2.0 may very well position the Liberty Alliance and SAML as competitors in the realm of identity federation.

5.2.2.3 Assessment

Table 5.4 Assessment of Liberty Alliance

Category	Information	Rating
Specification phase	• Liberty Alliance Architecture Overview Version 1.1: **Final**	**MEDIUM**

	• Liberty Identity Web Services Framework (ID-WSF) Overview Version 1.0-06: **Draft**	
Open standard	YES	**HIGH**
Potential to become open standard		**N/A**
Rate of advancement	Liberty Identity Web Services Framework (ID-WSF) Overview Version 1.0-06 publication date: **July 2003**	**HIGH**
Potential impact on Web Services		**HIGH**
Maturity level of consortium	2 years	**MEDIUM**
Number of implementations	19	**HIGH**

5.2.2.4 Implementations

The following URL provides a listing of current Liberty Alliance implementations:

- http://www.projectliberty.org/resources/enabled.html

5.2.2.5 Recommendation

Level 2: Emerging

There are a high number of Liberty Alliance implementations, and it has a large number of members (more than 160). However, we view the Liberty Alliance as a relatively immature consortium (2 years old). Once the Liberty Alliance consortium becomes more mature, its specifications may then be considered as “Level 1: Suitable For Use”.

We are skeptical of the general acceptance of the concept of federated network identities, due to the high risk of concentrating multiple attributes of an individual’s identity into a single entity. We also recommend that DISA monitor the advancement of SAML Version 2.0 and its potential overlap and competition with the Liberty Alliance specifications.

5.2.3 WS-Federation (GXA)

The WS-Federation specification is part of the Global XML Web Services Architecture (GXA). It was authored by Microsoft, IBM, Verisign, BEA Systems, and RSA Security. WS-Federation builds on WS-Trust to provide mechanisms for federated identity and security. In doing so, WS-Federation covers many of the same aspects that Liberty Alliance and OASIS SAML cover; such occurrences will be highlighted in this section. According to Microsoft, WS-Federation is "very complementary" to the Liberty Alliance's work in that Liberty Alliance “targets the specific scenario of consumers opting to allow their information to be shared among corporations or service providers, whereas WS-Federation addresses the broader issue of federating multiple identity systems to one another.”

5.2.3.1 Specification and Status

This section references the following specification:

- WS-Federation Version 1.0 (July 2003)

5.2.3.2 Main Concepts

WS-Federation Model

The WS-Federation Model is shown in the following figure:

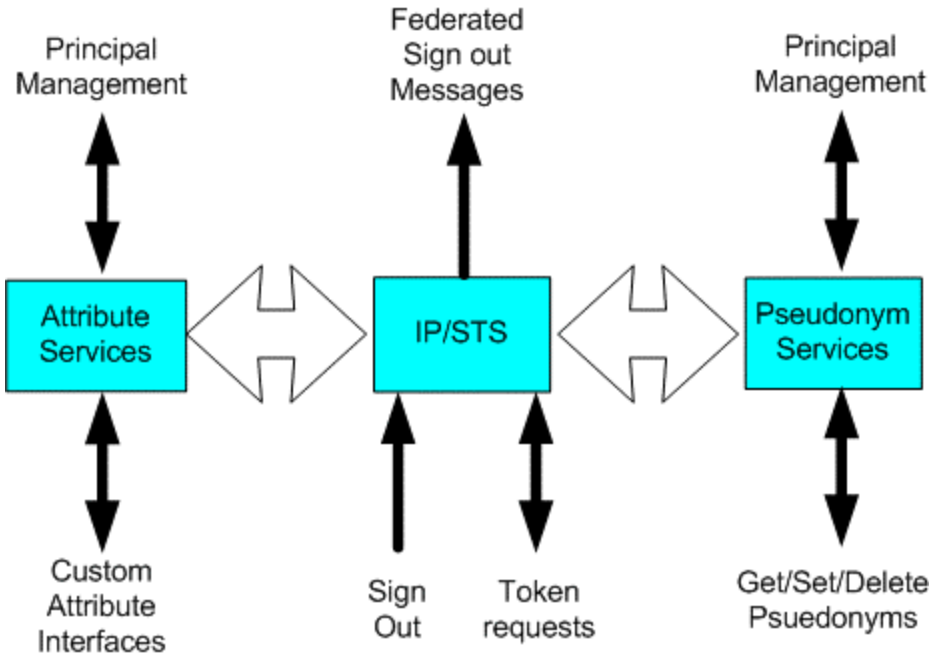

Figure 5.6 WS-Federation Model

Source: WS-Federation Version 1.0 Specification

There are three key components to this model:

- **Attribute Service:** Used to obtain authorized information about a principal (a system entity that may or may not represent a person). Similar in concept to SAML's attribute authority.
- **IP/STS (Identity Provider/Security Token Service):** Builds upon the WS-Trust notion of a *Security Token Service (STS)* to include identity management and authentication functions. The Identity Provider function is similar in concept to Liberty Alliance's identity service.
- **Pseudonym Service:** A Web Service that maintains alternate identity information about principals within a trust realm or federation.

These concepts are discussed in further detail below.

Virtual Security Domains/Single Sign-On (SSO)

WS-Federation allows a set of organizations to establish a single, virtual security domain. For example, a travel agent, an airline and a hotel chain may set up such a federation. This is similar in concept to Liberty Alliance's notion of federated network identity. Additionally, an end-user that "logs into" any member of the federation has effectively

logged into all of the members. This concept of *single sign-on* is also covered by both Liberty Alliance and SAML. The concept of a federation as a collection of realms that have established trust is similar to Liberty Alliance's concept of a circle of trust.

Simple Federation Scenario

The following figure illustrates a simple federation scenario in which security tokens from the requestor's trust realm are used to acquire security tokens from the resource's trust realm in order to access the resource/service:

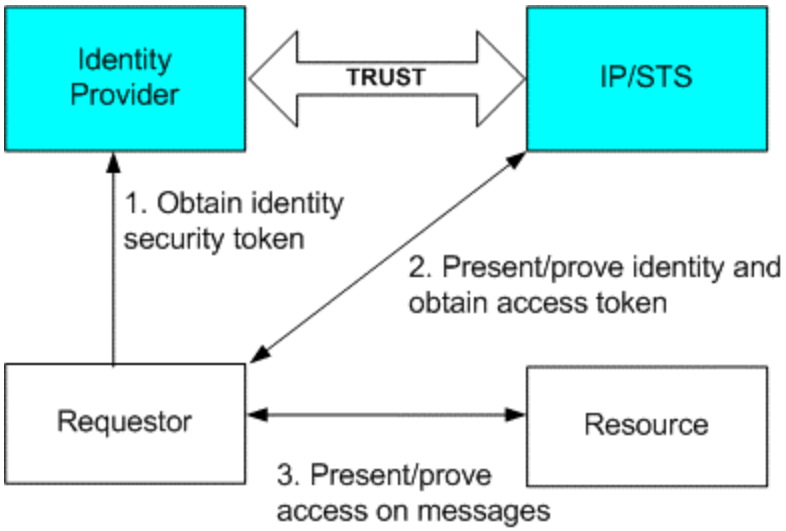

Figure 5.7 Using Security Tokens in WS-Federation

Source: WS-Federation Version 1.0 Specification

In the above example, a trust relationship has been established between the token services. An identity token is obtained by the first STS (step 1), and – because of the trust relationship – is accepted by the second STS (step 2) as proof of identity for obtaining an access token for the resource shown in the figure. Once the access token is obtained, the requestor can then access the resource (step 3). It is important to note that the resource may be a Web Service.

Such trust relationships also allow (for example) a token from one STS to be exchanged for another at a second STS, or possibly stamped or cross-certified by a second STS.

Additional Features

WS-Federation also contains a feature known as a *pseudonym service* that maintains alternate identity information ("aliases") about principals within a trust realm or federation. This allows a principal to have different aliases at different resources/services or in different realms, and to optionally have the pseudonym change per-service or per-login. This feature protects the privacy of the end-user and helps prevent access of end-user attributes by unauthorized parties.

WS-Federation also provides mechanisms for delegation – that is, to indicate that a requested or issued token should be delegated to another identity. The original requestor's policy indicates the degree of delegation it is willing to support.

Relation to Microsoft .NET Passport

Other than the fact that Microsoft is behind both initiatives, there is no direct relation between WS-Federation and Microsoft .NET Passport – although the two initiatives address generally the same functionality. Microsoft .NET Passport is one of the largest online authentication services in operation. Microsoft has constructed the Passport service to make it relatively easy for developers to build in Passport authentication to XML Web Services. The Passport model is similar in concept to Liberty Alliance's notion of a federated network identity, in which Microsoft "owns" all of the user's identity information. It is important to note that .NET Passport is not an open standard.

5.2.3.3 Assessment

Table 5.5 Assessment of WS-Federation

Category	Information	Rating
Specification phase	Initial public draft release	**LOW**
Open standard	NO	**LOW**
Potential to become open standard		**MEDIUM**
Rate of advancement	Publication date: July 2003	**HIGH**
Potential impact on Web Services		**HIGH**
Maturity level of consortium		**N/A**
Number of implementations	None	**LOW**

5.2.3.4 Implementations

None.

5.2.3.5 Recommendation

Level 3: Questionable

Although we believe that WS-Federation can have a high impact on Web Services, it is not being developed within an open standards consortium. However, The authors of WS-Federation have publicly pledged to submit the specification to a standards consortium. No decision has been made about which standards consortium, but, according to the authors, OASIS is a "very likely candidate." If the WS-Federation specification is ever transferred to an open standards consortium, it may then be considered as "Level 2: Emerging".

5.2.4 OASIS XML Common Biometric Format (XCBF)

Biometrics is the practice of verifying one's identity based on a physiological or behavioral characteristic, such as fingerprints, handwriting or retinal scans for the purpose of

recognizing the identity of an individual, or to verify a claimed identity. The OASIS XML Common Biometric Format (XCBF) specification defines cryptographic messages represented in XML markup for the secure collection, distribution, and processing, of biometric information. Mechanisms and techniques are described for the secure transmission, storage, and integrity and privacy protection of biometric data. XCBF can be used in applications as varied as homeland security, corporate privacy, law enforcement, and biotechnical research.

5.2.4.1 Specification and Status

This section references the following specifications:

- XML Common Biometric Format (XCBF) Version 1.1 (OASIS Standard, August 2003)
- WS-Security XCBF Token Profile (Working Draft 1.0, November 2002)

We will concentrate on the "XCBF Token Profile" specification in this section.

5.2.4.2 Main Concepts

XCBF and Web Services

The Web Services Security XCBF Token Profile describes how to use XML Common Biometric Format (XCBF) cryptographic messages within WS-Security. In this profile, a common XCBF security token is defined to convey and manage biometric information used for authentication and identification. The general processing model for WS-Security with XCBF objects is no different from that of other token formats described in WS-Security.

The following example illustrates an XCBF security token used with WS-Security:

```
[01] <?xml version="1.0" encoding="utf-8"?>
[02] <S:Envelope
[03]    <S:Header>
[04]       <wsse:Security>
[05]          <wsse:XCBFSecurityToken
[06]             wsu:Id="XCBF-biometric-object">
[07]             ValueType="wsse:XCBFv1"
[08]             EncodingType="wsse:XER">
[09]            <BiometricSyntaxSets>
[10]                <BiometricSyntax>
[11]                   <biometricObjects>
[12]                      <BiometricObject>
[13]                         <biometricHeader>
[14]                            <version>0</version>
[15]                            <recordType>
[16]                               <id>4</id>
[17]                            </recordType>
[18]                            <dataType>
[19]                               <processed/>
[20]                            </dataType>
[21]                            <purpose>
[22]                               <audit/>
[23]                            </purpose>
[24]                            <quality>80</quality>
[25]                            <validityPeriod>
[26]                               <notBefore> 1980.10.4 </notBefore>
```

```
[27]                              <notAfter>2003.10.3.23.59.59</notAfter>
[28]                            </validityPeriod>
[29]                            <format>
[30]                              <formatOwner>
[31]                                 <oid> 2.23.42.9.10.4.2 </oid>
[32]                              </formatOwner>
[33]                            </format>
[34]                         </biometricHeader>
[35]                         <biometricData>
[36]                            0A0B0C0D0E0F1A1B1C1D1E1F2A2B2C2D2E2F
[37]                         </biometricData>
[38]                      </BiometricObject>
[39]                   </biometricObjects>
[40]                </BiometricSyntax>
[41]             </BiometricSyntaxSets>
[42]          </wsse:XCBFSecurityToken>
[43]       </wsse:Security>
[44]    </S:Header>
[45]    <S:Body wsu:Id="MsgBody">  .....  </S:Body>
[46] </S:Envelope>
```

In the above example, the XCBF token is specified within a `<wsse:XCBFSecurityToken>` element [lines 05-42]. Information such as the following is specified:

- **recordType:** Identifier value of "4" indicates a "facial features" record [lines 15-17]
- **dataType:** Indicates that "processed" (versus "raw" or "intermediate") data is used [lines 18-20]
- **quality:** Indicates a data quality value of 80 (in the "excellent" range) [line 24]
- **biometricData:** A string of hexadecimal characters containing the actual biometric data [lines 35-37]

5.2.4.3 Assessment

Table 5.6 Assessment of XCBF

Category	Information	Rating
Specification phase	• XML Common Biometric Format (XCBF) Specification Version 1.1: **OASIS Standard** • WS-Security XCBF Token Profile: **Working Draft**	**MEDIUM**
Open standard	YES	**HIGH**
Potential to become open standard		**N/A**
Rate of advancement	Web Services Security XCBF Token Profile publication date: **November 2002**	**MEDIUM**
Potential impact on Web Services		**MEDIUM**
Maturity level of consortium	10 years	**HIGH**

Category	Information	Rating
Number of implementations	None	**LOW**

5.2.4.4 Implementations

None

5.2.4.5 Recommendation

Level 3: Questionable

The main reason for this recommendation level is the lack of advancement of the Web Services Security XCBF Token Profile; additionally, there are no identified implementations of this profile. If this profile is advanced in the near future and some implementations are announced, it may then be considered as “Level 2: Emerging”.

5.3 Integrity/Non-Repudiation

Integrity involves ensuring that a message or document that is digitally signed has not been changed or corrupted in transit. *Non-repudiation* is the ability to ensure that a party to a contract or communication cannot deny the authenticity of their signature on a document or the sending of a message that they originated. Because these security functional categories are closely related, we discuss them in the same section.

WS-Security provides message-level integrity and non-repudiation. However, this functionality is also required at the session level. The Web Services Secure Conversation Language (WS-SecureConversation) specification provides session-level integrity and non-repudiation. WS-SecureConversation also provides session-level confidentiality.

The following specification is discussed in this section:

- Web Services Secure Conversation Language (WS-SecureConversation)

5.3.1 Web Services Secure Conversation Language (WS-SecureConversation)

The OASIS WS-Security specification readily supports Web Service scenarios that involve only the short sporadic exchange of a few messages. It also supports scenarios that involve long- duration, multi-message conversations between the Web Services. However, WS-Security’s solution for this second type of scenario is not optimal for several reasons:

- Its repeated use of computationally expensive cryptographic operations such as public key validation
- Sending and receiving many messages using the same cryptographic keys allows brute force attacks that "break the code”

It is for these very reasons that protocols such as HTTP/S use public keys to perform a simple negotiation that defines conversation-specific keys. This key exchange allows more efficient security implementations and also decreases the amount of information encrypted with a specific set of keys.

The WS-SecureConversation specification provides similar support for WS-Security. WS-Security can be used with public keys to start a "conversation" or "session," and WS-SecureConversation can then be used to agree on session-specific keys for signing and encrypting information. The WS-SecureConversation specification is part of the Global XML Web Services Architecture (GXA). It was authored by Microsoft, IBM, Verisign, and RSA Security.

5.3.1.1 Specification and Status

This section references the following specification:

- WS-SecureConversation Version 1.0 (December 2002).

5.3.1.2 Main Concepts

Security Context Token

The main entity in WS-SecureConversation is a *security context token*. A security context token is a token that is used by both parties in a multi-message exchange as part of an established security context—it is also referred to as a *"shared secret"*. The lifetime of a security context token extends throughout the communications session, after which it ceases to exist—hence the tighter security advantage over the message authentication model of WS-Security.

The following example illustrates the use of a security context token in establishing a security context:

```
[01] <wsse:Security>
[02]     <wsse:SecurityContextToken>
[03]         <wsu:Identifier>http://securitycontextid.com</wsu:Identifier>
[04]         <wsu:Expires>2002-08-31T13:20:00-05:00</wsu:Expires>
[05]         <wsse:Keys>
[06]            <xenc:EncryptedKey Id="newSharedSecret">
[07]                ...
[08]            </xenc:EncryptedKey>
[09]         </wsse:Keys>
[10]     </wsse:SecurityContextToken>
[11] </wsse:Security>
```

The above specifies a security context identifier [line 03] and key that is used as the shared secret [lines 06-08].

Establishing Security Context

WS-SecureConversation presents three ways in which a security context can be established:

- The security token is created by a security token service, as defined in WS-Trust
- The security token is created by one of the communicating parties, and propagated with a message
- The security token is created through negotiation between participants

Derived Keys

WS-SecureConversation also specifies the use of *derived keys* in a multi-message exchange. Derived keys further enhance security by allowing a different key to be used for

each exchange between participants, thereby eliminating the need to store a particular key. Each successive key is derived from a key that was used on a previous exchange within the communications session using an algorithm defined in WS-SecureConversation.

The following example illustrates a derived key:

```
[01] <DerivedKeyToken>
[02]     <SecurityTokenReference>
[03]                 ...
[04]     </SecurityTokenReference>
[05]     <Generation>4</Generation>
[06] </DerivedKeyToken>
```

In the above example, the derived key is based on the 5th generation [line 05] of the shared secret (note that generations start with 0).

5.3.1.3 Assessment

Table 5.7 Assessment of WS-SecureConversation

Category	Information	Rating
Specification phase	Initial public draft release	**LOW**
Open standard	NO	**LOW**
Potential to become open standard		**MEDIUM**
Rate of advancement	Publication date: December 2002	**MEDIUM**
Potential impact on Web Services		**HIGH**
Maturity level of consortium		**N/A**
Number of implementations	1	**LOW**

5.3.1.4 Implementations

- Microsoft Web Services Enhancements (WSE): http://msdn.microsoft.com/webservices/building/wse/default.aspx

5.3.1.5 Recommendation

Level 3: Questionable

Although we believe that WS-SecureConversation can have a high impact on Web Services, it is not being developed within an open standards consortium. If the WS-SecureConversation specification is ever transferred to an open standards consortium, it may then be considered as “Level 2: Emerging”.

5.4 Confidentiality

Confidentiality involves protecting information from interception during transmission. This security functionality category is covered by WS-Security (message-level confidentiality) and WS-SecureConversation (session-level confidentiality), both of which have already been discussed.

5.5 Trust

Trust can be defined as “the characteristic that one entity is willing to rely upon a second entity to execute a set of actions and/or to make set of assertions about a set of subjects and/or scopes”[1].

The following specifications are discussed in this section:

- Web Services Trust Language (WS-Trust)

5.5.1 W3C Signature

5.5.2 WS-Trust (GXA)

The WS-Trust specification is part of the Global XML Web Services Architecture (GXA). It was authored by Microsoft, IBM, Verisign, and RSA Security. WS-Trust defines an extensible model for setting up and verifying trust relationships, allowing Web Services to agree on which security servers they “trust” and to rely on these servers.

5.5.2.1 Specification and Status

This section references the following specification:

- WS-Trust Version 1.0 (December 2002)

5.5.2.2 Main Concepts

Trust Engine/Security Token Service

In order to secure a communication between two parties, the two parties must exchange security credentials (either directly or indirectly). However, each party needs to determine if they can "trust" the asserted credentials of the other party. WS-Trust introduces the notion of a *trust engine*, a conceptual component of a Web Service that evaluates the security-related aspects of a message. A trust engine verifies that:

- The claims in a security token are sufficient to comply with the policy and that the message conforms to the policy
- The attributes of the claimant are proven by the signatures
- The issuers of the security tokens are trusted to issue the claims they have made

For example, if a policy stated that only Kerberos tickets were accepted as a security token, the trust engine of a Web Service would enforce this requirement for all incoming messages.

WS-Trust also introduces the notion of a security token service that issues security tokens based on trust, similar to a *certificate authority (CA).* The following figure illustrates a "sender" and "receiver" Web Service, each with its own trust engine. A *security token service* is also depicted, from which the sender Web Service will request a security token to be used for its interaction with the receiver Web Service. The sender Web Service can request a security token based on the receiver Web Service's policies, using the mecha-

[1] Source: WS-Trust Version 1.0 Specification.

nisms described earlier in this article. The sender Web Service will use its trust engine to authenticate the security token service, while the receiver Web Service will use its trust engine to authenticate the sender Web Service.

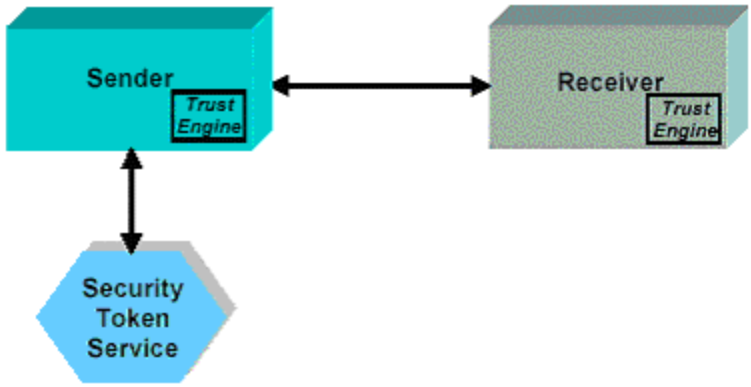

Figure 5.8 Use of Trust Engines in WS-Trust

The following example demonstrates a request for a security token (X.509 certificate), and the response with the certificate:

```
[01] <wsse:RequestSecurityToken>
[02]     <wsse:TokenType>wsse:X509v3</wsse:TokenType>
[03]     <wsse:RequestType>wsse:ReqIssue</wsse:RequestType>
[04] </wsse:RequestSecurityToken>
[05] <wsse:RequestSecurityTokenResponse>
[06]     <wsse:RequestedSecurityToken>
[07]         <BinarySecurityToken ValueType="wsse:X509v3"
[08]                              EncodingType="wsse:Base64Binary">
[09]                              MIIEZzCCA9CgAwIBAgIQEmtJZc0...
[10]         </BinarySecurityToken>
[11]     </wsse:RequestedSecurityToken>
[12] </wsse:RequestSecurityTokenResponse>
```

In the above example, the "ReqIssue" value [line 03] denotes the issuance of a security token. Other valid values are "ReqValidate" (validate security token) and "ReqExchange" (exchange security token).

The following figure illustrates this request/response interaction:

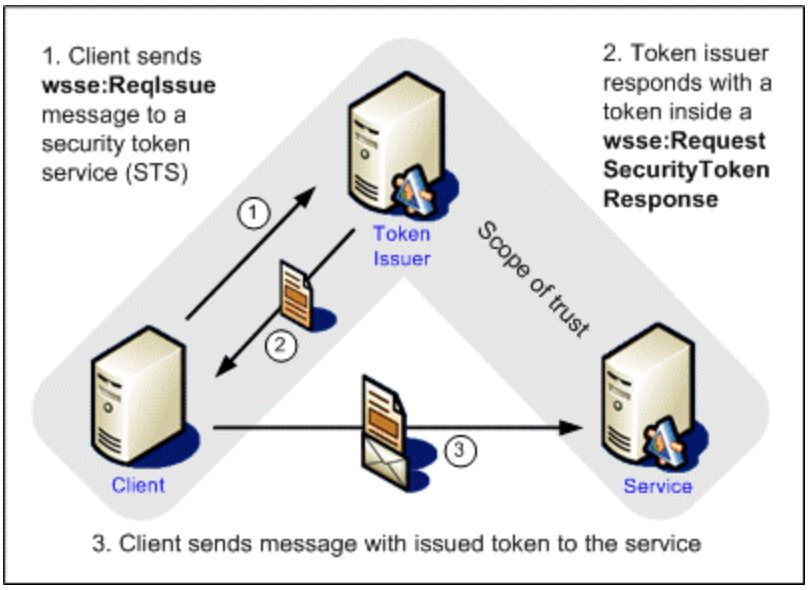

Figure 5.9 WS-Trust Interactions

Source: WS-Trust
Version 1.0 Specification

In some cases, a security token service may choose to challenge the requestor of a security token. For example, this may occur if the security token service does not trust the nonce and timestamp (for example, the freshness) in the message. Or, the security token service may challenge the signature within the message.

5.5.2.3 Assessment

Table 5.8 Assessment of WS-Trust

Category	Information	Rating
Specification phase	Initial public draft release	**LOW**
Open standard	NO	**LOW**
Potential to become open standard		**MEDIUM**
Rate of advancement	Publication date: December 2002	**MEDIUM**
Potential impact on Web Services		**HIGH**
Maturity level of consortium		**N/A**
Number of implementations	1	**LOW**

5.5.2.4 Implementations

- Microsoft Web Services Enhancements (WSE): http://msdn.microsoft.com/webservices/building/wse/default.aspx

5.5.2.5 Recommendation

Level 3: Questionable

Although we believe that WS-Trust can have a high impact on Web Services, it is not being developed within an open standards consortium. If the WS-Trust specification is ever transferred to an open standards consortium, it should be considered as "Level 2: Emerging".

5.6 Authorization/Policy

Authorization involves controlling access privileges to resources. A *policy* is set of information that conveys various characteristics of, and rules for, a Web Service, such as authorization and the types of security tokens that a Web Service accepts. Because these security functional categories are closely related, we discuss them in the same section.

The following specifications are discussed in this section:

- OASIS XACML
- W3C Open Digital Rights Language (ODRL)
- OASIS Extensible Rights Markup Language (XrML)
- Web Services Policy Framework (WS-Policy)

5.6.1 OASIS eXtensible Access Control Markup Language (XACML)

OASIS eXtensible Access Control Markup Language (XACML) is an XML specification for expressing policies for information access over the Internet. It is being developed by the OASIS Access Control Markup Language (XACML) TC). In most cases today, organizations have widely dispersed security policies, with elements managed by different departments. Consequently, it is very difficult to obtain an aggregated view of an organization's access control policy, as well as modify it as necessary. XACML's common language for expressing security policies allows an enterprise to efficiently manage the enforcement of its security policy in all the components of its information systems.

5.6.1.1 Specification and Status

This section references the following specifications:

- XACML Version 1.0 (OASIS Standard, February 2003)
- XACML Profile for Web Services (Working Draft 04, September 2003)

The XACML Version 1.1 specification is currently in process.

5.6.1.2 Main Concepts

Subject/Resource/Action

XACML is based on three main concepts:

- Subject: An entity (human or system) that requests access to a resource
- Resource: A data, service, or system component to which access is requested

- Action: An operation on a resource (such as “read”)

In XACML, a *subject* requests access to a *resource* to perform some *action* on that resource.

Policies and Rules

A policy in XACML is essentially a set of rules that form the basis for an *authorization decision*. An example of a policy is “Any user with an e-mail name in the XYZ Corporation domain is allowed to perform any action on any resource”.

An authorization decision is the result of a *decision request*. An example of a decision request is “User John Smith with e-mail name jsmith@abc.com would like to read financial records from the year 2002 at XYZ Corporation”. A decision request evaluates to “Permit” (i.e. permit access to a resource), “Deny” (i.e. deny access to a resource), “Indeterminate” (i.e. a decision could not be determined), or “Not Applicable” (i.e. there is no policy that applies to the request). An applicable policy for a decision request is located based on the subject, resource, and action values in the decision request.

Policies are comprised of *rules* that are individual evaluated and whose evaluation results are combined to arrive at an authorization decision.

Policy Processing

The following system entities are defined by XACML for its policy processing:

- Policy Access Point (PAP): Creates policies and makes them available to Policy Decision Points (PDPs)
- Policy Decision Point (PDP): Evaluates applicable policies and renders authorization decisions
- Policy Enforcement Point (PEP): Performs access control by making decision requests and enforcing authorization decisions
- Policy Information Point (PIP): Acts as a source of attribute values
- Authorization decision statement: Expresses whether a given subject has been granted specific permissions to access a particular resource

XACML and Web Services

The “XACML Profile for Web Services” defines mechanisms for enforcing access control to a Web Service endpoint, as well as expressing policies in areas such as reliable messaging, privacy, trust, authentication, and cryptographic security. Policies are associated with Web Service endpoint definitions, with WSDL 1.1 ports are identified as resources. Policies may also be targeted more finely than ports (i.e. to operations and messages).

The following is an example of XACML policy used in conjunction with a Web Service. It references an online book club, and allows only members of the book club to order items online from that book club:

```
[01] <?xml version=1.0" encoding="UTF-8"?>
[02] <Policy xmlns="urn:oasis:names:tc:xacml:1.0:policy"
[03]     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

```
[04]     xsi:schemaLocation="urn:oasis:names:tc:xacml:1.0:policy
[05]     http://www.oasis-open.org/tc/xacml/1.0/cs-xacml-schema-policy-01.xsd"
[06]     PolicyId="identifier:example:SimplePolicy1"
[07]     RuleCombiningAlgId="identifier:rule-combining-algorithm:deny-
         overrides">
[08]    <Description>
[09]       XYZ Book Club “order” access control policy
[10]     </Description>
[11]     <Target>
[12]          <Subjects>
[13]              <AnySubject/>
[14]          </Subjects>
[15]          <Resources>
[16]              <ResourceMatch MatchId=”equal”
[17]                  <AttributeValue
DataType=”anyURI”>serviceX:portX</AttributeValue>
[18]                  <ResourceAttributeDesignator AttributeID=
[19]                      ”urn:oasis:names:tc:xacml:1.0:attribute:portId
                                                DataType=”anyURI”/>
[20]             </ResourceMatch>
[21]          </Resources>
[22]          <Actions>
[23]              <AnyAction/>
[24]          </Actions>
[25]     </Target>
[26]     <Rule Effect="Permit">
[27]         <Description>
[28]             Only members of XYZ Book Club can place orders.
[29]         </Description>
[30]         <Condition FunctionId="and">
[31]              <Apply FunctionId="equal">
[32]                  <AttributeValue>member</AttributeValue>
[33]                  <SubjectAttributeDesignator AttributeId="membership-
status"/>
[34]              </Apply>
[35]              <Apply FunctionId="equal">
[36]                  <AttributeValue>order</AttributeValue>
[37]                  <ActionAttributeDesignator AttributeId="action-id"/>
[38]              </Apply>
[39]          </Condition>
[40]     </Rule>
[41] </Policy>
```

The above policy is applicable to any subject [lines 12-14] and any action [lines 22-24]. The resource target is the port whose portId is “serviceX:portX” [line 17]. The policy contains a single rule [lines 26-40], which states that the effect of the rule will be “Permit” [line 26] only if the requestor has a membership status of “member” [lines 31-34] and the requested action was “order” [lines 35-38].

XACML and SAML

XACML and SAML can be used in conjunction for authentication (SAML) and authorization (XACML) in Web Services exchanges. For example, a SAML assertion can be evaluated by a Web Service’s authentication mechanism as a first step to determining that the subject that is making a request is who they claim to be. Once the subject is authenticated, XACML would be invoked to check for the authenticated subject’s authorization to perform the requested action.

We foresee a greater degree of interoperability between SAML and XACML in the future. Consideration is being given for the SAML Version 2.0 specification to include

mechanisms for issuing XACML requests and handling XACML responses, as well as a potentially greater alignment between formats of subjects between SAML and XACML.

5.6.1.3 Assessment

Table 5.9 Assessment of XACML

Category	Information	Rating
Specification phase	• XACML Version 1.0: **OASIS Standard** • XACML Profile for Web Services: **Working Draft**	**MEDIUM**
Open standard	YES	**HIGH**
Potential to become open standard		**N/A**
Rate of advancement		**N/A**
Potential impact on Web Services		**HIGH**
Maturity level of consortium	10 years	**HIGH**
Number of implementations	2	**MEDIUM**

5.6.1.4 Implementations

The following URL provides a listing of current XACML implementations:

- http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml

5.6.1.5 Recommendation

Level 2: Emerging

We believe that the functionality that XACML provides (access control for Web Services) is greatly needed, and we foresee the specification as having a very high impact on the advancement of Web Services. The main reason for this recommendation level is the low level of XACML implementations currently available. If more implementations emerge, it may then be considered for advancement to “Level 1: Suitable For Use”.

5.6.2 W3C Open Digital Rights Language (ODRL)

The W3C Open Digital Rights Language (ODRL) is an XML-based language for *digital rights management (DRM)*. DRM focuses on enabling secure distribution - and perhaps more importantly, to disable illegal distribution - of paid content over the Web. Several of the concepts covered by ODRL can also be applied to Web Services.

5.6.2.1 Specification and Status

This section references the following specification:

- Open Digital Rights Language (ODRL) Version 1.1 (W3C Note, September 2002)

5.6.2.2 Main Concepts

ODRL Foundation Model

DRM covers the digital management of rights - be they rights in a physical manifestation of a work (e.g. a book), or rights in a digital manifestation of a work (e.g. an e-book). ODRL is a standard language and vocabulary for the expression of terms and conditions over assets.

The ODRL Foundation Model illustrates the main entities represented in the ODRL specification, and the relationships between them. It is shown in the following figure:

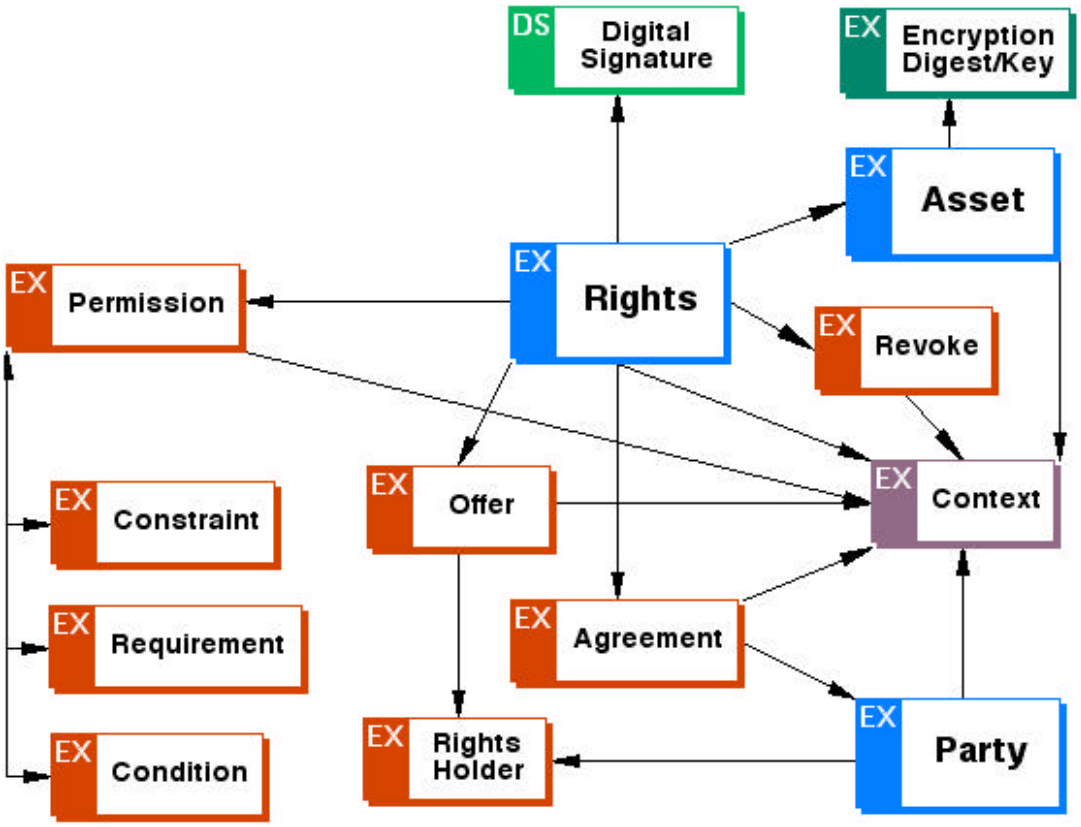

Figure 5.10 ODRL Foundational Model

Source: ODRL Version 1.1 Specification

This model consists of the following three *core entities:*

- Assets
- Rights
- Parties

An *asset* is any physical or digital content, such as a Web Service. *Rights* encompass usage rights for an asset, and include *permissions*. An example of a permission is “play a video asset”. *Constraints* are used to limit permissions–for example, “play the video for a maximum of 5 times”. *Requirements* are the obligations needed to exercise a permission–for example, “play $5 each time you play the video”. *Conditions* specify exceptions that,

if they become true, expire the permissions and possibly require renegotiation–for example, “if credit card expires, then all permissions are withdrawn to play the video”.

A *party* may be an end user or a rights holder. *End users* are usually the asset consumers, while *rights holders* are usually parties that have played some role in the creation, production, distribution of the asset and can assert some form of ownership over the asset and/or its permissions. Rights holders may also receive royalties.

Finally, *offers* and *agreements* are the recognized set of activities allowed over an asset.

ODRL and Web Services

Since ODRL identifies a digital asset via its URI, one can describe the rights over Web Services using ODRL (i.e. to enforce access control). This is shown in the following example:

```
[01] <agreement>
[02]     <party>
[03]         <context>
[04]             <uid>urn:renato.iannella</uid>
[05]         </context>
[06]        <rightsholder/>
[07]     </party>
[08]     <asset>
[09]         <context>
[10]            <uid>http://example.com/my-web-service</uid>
[11]        </context>
[12]     </asset>
[13]    <permission>
[14]         <execute/>
[15]    </permission>
[16]    <party>
[17]        <context>
[18]            <uid>urn:people:jsmith</uid>
[19]         </context>
[20]       <rightsholder/>
[21]    </party>
[22] </agreement>
```

In the above example, a user “jsmith” [line 18] has been granted permission to "execute" a Web Service [lines 13-15] identified by URL http://example.com/my-web-service [line 10].

Future Releases

The authors of ODRL are planning new features for version 2.0 for early 2004. However, it is not clear at this time if W3C will advance DRM specifications.

5.6.2.3 Assessment

Table 5.10 Assessment of ODRL

Category	Information	Rating
Specification phase	W3C Note	**LOW**
Open standard	YES	**HIGH**
Potential to become open standard		**N/A**

Category	Information	Rating
Rate of advancement	Publication date: **September 2002**	**LOW**
Potential impact on Web Services		**HIGH**
Maturity level of consortium	9 years	**HIGH**
Number of implementations	None	**LOW**

5.6.2.4 Implementations

While vendor implementations of ODRL exist, no current vendor implementations of ODRL focusing on use in conjunction with Web Services have been identified.

5.6.2.5 Recommendation

Level 3: Questionable

The main reasons for this recommendation level are the lack of clear direction for ODRL within W3C, and the fact that the access control capability for Web Services described earlier can be carried out using other more advanced open standards (e.g. OASIS XACML). Additionally, we do not foresee a need in the medium-term future for other ODRL capabilities (such as payment) for Web Services. We therefore believe that this specification may shift to "Level 4: Do Not Use" within the next year.

5.6.3 OASIS eXtensible Rights Markup Language (XrML)

The Extensible Rights Markup Language (XrML) provides a universal method for securely specifying and managing rights and conditions associated with all kinds of resources including digital content as well as services. XrML is being developed by the OASIS Rights Language TC. XrML has its roots in the Xerox Palo Alto Research Center (PARC), when it was first introduced in 1996 as Digital Property Rights (DPR). Using XrML, anyone owning or distributing digital resources can specify and identify the parties allowed to use those resources, the rights available to those parties, and the terms and conditions under which those rights may be exercised.

XrML covers many of the same aspects that the W3C Open Digital Rights Language (ODRL) covers; such occurrences will be highlighted in this section.

5.6.3.1 Specification and Status

This section references the following specification:

- Extensible Rights Markup Language (XrML) Version 2.1 Technical Overview (Draft, May 2002)

5.6.3.2 Main Concepts

Core Elements

There are four *core elements* in XrML:

- Principal

- Right
- Resource
- Condition

These four elements comprise a *grant*. A *principal* encapsulates the identification of a party to whom rights are granted; it is similar in concept to a party in ODRL. A *right* is the "verb" that a principal can be granted to exercise against some resource under some condition; it typically specifies an action or a class of actions that a principal may perform on or using the associated resource. It is similar in concept to a right in ODRL. A *resource* is the "object" to which a principal can be granted a right; it is similar in concept to an asset in ODRL. Finally, a *condition* specifies the terms, conditions, and obligations under which rights can be exercised; it is similar in concept to a condition in ODRL.

License

The central XrML construct is a *license*. A license is conceptually a container of grants, each of which conveys to a particular principal the sanction to exercise some identified right against some identified resource, and possibly subject to the need for some condition to first be fulfilled.

The following figure illustrates these entities and their relationships within the XrML data model:

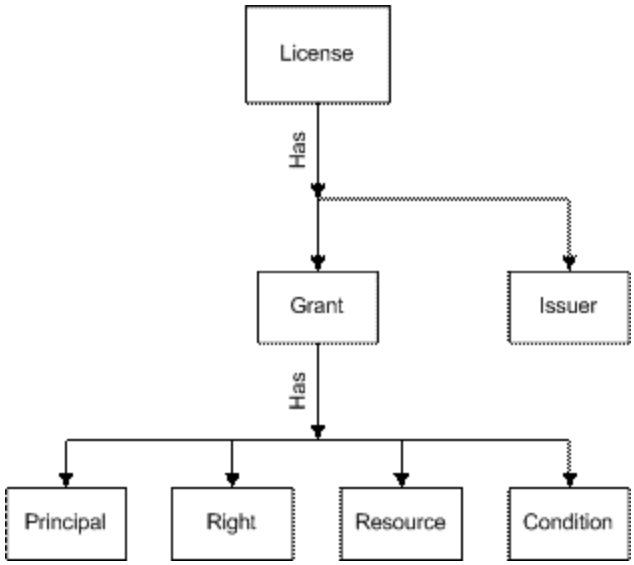

Figure 5.11 XrML Concepts and Relationships

Source: ContentGuard

XrML and Web Services

The OASIS WS-Security TC is in the process of completing an XrML profile that describes the carrying of XrML tokens in a WS-Security header.

5.6.3.3 Assessment

Table 5.11 Assessment of XrML

Category	Information	Rating
Specification phase	OASIS Draft	**LOW**
Open standard	YES	**HIGH**
Potential to become open standard		**N/A**
Rate of advancement	Publication date: **May 2002**	**LOW**
Potential impact on Web Services		**HIGH**
Maturity level of consortium	10 years	**HIGH**
Number of implementations	None	**LOW**

5.6.3.4 Implementations

While no current vendor implementations of XrML focusing on use in conjunction with Web Services have been identified, the following companies have publicly announced their active support and agreement to build products and/or services which are XrML compliant: Microsoft, OverDrive, Zinio Systems, DMDsecure, Integrated Management Concepts and Content Works.

5.6.3.5 Recommendation

Level 3: Questionable

The main reasons for this recommendation level are the slow rate of advancement of the XrML specification within OASIS, and the fact that (as with ODRL) the XrML functionality most applicable to Web Services can be carried out by other more advanced open standards (e.g. OASIS XACML). We therefore believe that this specification may shift to "Level 4: Do Not Use" within the next year.

5.6.4 Web Services Policy Framework (WS-Policy)

The Web Services Policy Framework (WS-Policy) is part of the Global XML Web Services Architecture (GXA). The latest version was released in May 2003 by Microsoft, IBM, Verisign, BEA Systems, and SAP. WS-Policy provides a general-purpose model for describing and communicating the policies of a Web Service.

5.6.4.1 Specification and Status

This section references the following specifications:

- WS-Policy Version 1.1 (May 2003)
- WS-PolicyAssertions Version 1.1 (May 2003)
- WS-PolicyAttachment Version 1.1 (May 2003)
- WS-SecurityPolicy Version 1.0 (December 2002)

5.6.4.2 Main Concepts

Policy Assertions

WS-Policy defines a policy to be a collection of one or more *policy assertions*. A policy assertion conveys a requirement, preference, or capability of a given policy subject. Some policy assertions specify traditional requirements and capabilities that will ultimately manifest on the wire (e.g., authentication scheme, transport protocol selection), while some specify requirements and capabilities that have no wire manifestation yet are critical to proper service selection and usage (e.g., privacy policy, QoS characteristics).

Policy Expressions

A *policy expression* is an XML representation of a policy. The following example illustrates a policy expression that uses "SecurityToken" assertions to specify the security token types required/accepted by a Web Service:

```
[01] <wsp:Policy>
[02]   <wsp:ExactlyOne>
[03]     <wsse:SecurityToken TokenType="wsse:X509v3"
[04]           wsp:Usage="wsp:Required" wsp:Preference="50"/>
[05]     <wsse:SecurityToken TokenType="wsse:Kerberosv5TGT"
[06]           wsp:Usage="wsp:Required" wsp:Preference="10"/>
[07]   </wsp:ExactlyOne>
[08] </wsp:Policy>
```

The above policy states that the Web Service with which the policy is associated can accept either [line 02] X.509 certificates [lines 03-04] or Kerberos tickets [lines 05-06] for authentication. Additionally, the "preference" [lines 04 and 06] is that an X.509 certificate is used for authentication, due to its higher "Preference" value of 50 [line 04].

Related Specifications

There are several other policy-related GXA specifications; several of these are used in conjunction with WS-Policy. These are:

- **WS-PolicyAssertions:** Describes general messaging-related policy assertions such as character encodings (i.e. the character encodings supported by a Web Service), spoken languages (i.e. the spoken languages supported by a Web Service), and specification versions (i.e. which versions of a specification a Web Service supports)
- **WS-PolicyAttachment:** Specifies various attachment mechanisms for using policy expressions with existing Web Services technologies, such as how to associate policy expressions with WSDL type definitions and UDDI entities
- **WS-SecurityPolicy:** Defines how to describe policies related to various features defined WS-Security, such as accepted security token types, or what portions of a message must be signed or encrypted

WS-Policy and XACML

Although WS-Policy and XACML overlap in some areas of functionality, there are actually large inherent differences between the two specifications. WS-Policy is a more general language that can be used to describe properties and capabilities of many different

types of resources, including Web Services and Web Services endpoints, while XACML is specifically an access control rule language. Although XACML can be used to specify some of the non-access-control of policies that WS-Policy specifies, it is not inherently meant to do so. Additionally, WS-Policy is not intended to be interpreted (in the sense of programming language execution) but rather to be processed as data from which useful information can be extracted. The principal usage of XACML, however, is to be consumed by an XACML rule evaluation engine at an access control decision point for making access control decisions.

5.6.4.3 Assessment

Table 5.12 Assessment of WS-Policy

Category	Information	Rating
Specification phase	Initial public draft release (all)	**LOW**
Open standard	NO	**LOW**
Potential to become open standard		**MEDIUM**
Rate of advancement	• WS-Policy Version 1.1 publication date: **May 2003** • WS-PolicyAssertions Version 1.1 publication date: **May 2003** • WS-PolicyAttachment Version 1.1 publication date: **May 2003** • WS-SecurityPolicy Version 1.0 publication date: **December 2002**	**HIGH**
Potential impact on Web Services		**HIGH**
Maturity level of consortium		**N/A**
Number of implementations	None	**LOW**

5.6.4.4 Implementations

None.

5.6.4.5 Recommendation

Level 3: Questionable

Although we believe that WS-Policy can have a high impact on Web Services, it is not being developed within an open standards consortium. However, The authors of WS-Policy have publicly announced intentions to submit the specification to a standards consortium. If the WS-Policy specification is ever transferred to an open standards consortium, it may then be considered as “Level 2: Emerging”.

5.7 General Recommendations

A large number of open standards (and potential open standards) are currently emerging in the realm of security. We foresee the current lack of overall robust security for Web Services improving greatly in the next two years, as many of the specifications discussed in this section mature and others arise. The current lack of overall robust security makes it difficult to execute Web Services scenarios that stretch beyond "point-to-point" interactions; we therefore recommend that DISA utilize Web Services at this time, but in point-to-point interactions using established mechanisms such as traditional Public Key Infrastructure (PKI) and Secure Socket Layer/Transport Layer Security (SSL/TLS).

The following is a summary of the recommendations given in this section:

- We believe that WS-Security will have the largest single impact on the advancement of Web Services of any of the specifications discussed in this section. If WS-Security becomes an OASIS standard and more implementations emerge, it current Level 2 status may then be considered for upgrade to "Level 1: Suitable For Use"
- We recommend that OASIS SAML be used at this time, as it has gained wide acceptance in many different industries and settings. We foresee the number of SAML implementations growing steadily in the medium- and long-term future.
- We view the Liberty Alliance as a relatively immature consortium (2 years old). Once the Liberty Alliance consortium becomes more mature, its current Level 2 status may then be considered for upgrade to "Level 1: Suitable For Use".
- We believe that the various GXA specifications discussed in this section can have a high impact on Web Services. However, with the exception of WS-Security, none of the GXA specifications have been advanced into open standards consortiums. If this occurs, the current Level 3 status for each GXA specification may then be considered for upgrade to "Level 2: Emerging".
- We believe that the functionality that XACML provides (access control for Web Services) is greatly needed, and we foresee the specification as having a very high impact on the advancement of Web Services. However, the current lack of implementations leads XACML to be evaluated here as "Level 2: Emerging".
- We do not foresee the emerging Digital Rights Management (DRM) specifications as being highly valuable to Web Services at this time.

5.8 References

WS-Security: SOAP Message Security:
http://www.oasis-open.org/committees/download.php/3281/WSS-SOAPMessageSecurity-17-082703-merged.pdf

WS-Security: Username Token Profile:
http://www.oasis-open.org/committees/download.php/3154/WSS-Username-04-081103-merged.pdf

WS-Security: X.509 Token Profile:
http://www.oasis-open.org/committees/download.php/3214/WSS-X509%20draft%2010.pdf

Joseph M. Chiusano, "Web Services Security and More: The Global XML Web Services Architecture (GXA)", Developer.com, March 2003
Joseph M. Chiusano, "Web Services Security and More Part 2: The Global XML Web Services Architecture (GXA)", Developer.com, May 2003

OASIS SAML Version 1.1:
http://www.oasis-open.org/committees/download.php/2949/sstc-saml-1.1-cs-03-pdf-xsd.zip

Seshardri Gokul, "Authenticating Web Services with SAML", informIT.com, August 2002

Liberty Alliance Architecture Overview Version 1.1:
http://www.projectliberty.org/specs/archive/v1_1/liberty-architecture-overview-v1.1.pdf

Liberty Identity Web Services Framework (ID-WSF) Overview Version 1.2-06:
http://www.projectliberty.org/specs/draft-lib-arch-idwsf-overview-v1.2-06.pdf

WS-Federation Version 1.0:
http://msdn.microsoft.com/library/en-us/dnglobspec/html/ws-federation.asp

OASIS XML Common Biometric Format (XCBF) Version 1.1:
http://www.oasis-open.org/apps/org/workgroup/xcbf/download.php/3353/oasis-200305-xcbf-specification-1.1.doc

WS-Security XCBF Token Profile:
http://www.oasis-open.org/committees/wss/documents/WSS-XCBF.doc

WS-SecureConversation Version 1.0:
http://msdn.microsoft.com/library/en-us/dnglobspec/html/ws-secureconversation.asp

WS-Trust Version 1.0:
http://msdn.microsoft.com/library/en-us/dnglobspec/html/ws-trust.asp

OASIS XACML Version 1.0:
http://www.oasis-open.org/committees/download.php/2406/oasis-xacml-1.0.pdf

OASIS XACML Profile for Web Services:
http://www.oasis-open.org/committees/download.php/3661/draft-xacml-wspl-04.pdf

W3C Open Digital Rights Language (ODRL) Version 1.1:
http://www.w3.org/TR/odrl/

Extensible Rights Markup Language (XrML) Version 2.1 Technical Overview:
http://xml.coverpages.org/XrMLTechnicalOverview21-DRAFT.pdf

WS-Policy Version 1.1:
http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnglobspec/html/ws-policy.asp

WS-PolicyAssertions Version 1.1:
http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnglobspec/html/ws-policyassertions.asp

WS-PolicyAttachment Version 1.1:
http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnglobspec/html/ws-policyattachment.asp

WS-SecurityPolicy Version 1.0:
http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnglobspec/html/ws-securitypolicy.asp

Secure, Reliable, Transacted Web Services: Architecture and Composition:
http://msdn.microsoft.com/webservices/understanding/advancedwebservices/default.aspx?pull=/library/en-us/dnwebsrv/html/wsoverview.asp

Microsoft Passport and the Future of Authentication:
http://www.technewsworld.com/perl/story/31667.html

6 Interoperability of Web Services

Interoperability is one of the pillars on which Web Services are built. The goal is to allow different Web Services to be able to work together seamlessly. A system is going to search for a particular Web Service and choose between different implementations to leverage. This is the reason behind the vast number of specifications, to achieve this level of interoperability.

One concern is that due to the number of standards organizations and the number of specifications being submitted, some specifications may contain contradictions or ambiguities that make interoperability between two specifications difficult. Another concern is that a specification can contain loosely defined requirements, optional features, or ambiguities that could cause interoperability issues between two implementations of the given specification.

The Web Services Interoperability Organization (WS-I) (http://www.ws-i.org) was formed to address these concerns. Currently over 170 organizations have joined WS-I. WS-I is not a standards body trying to produce new specifications, but instead seeks to clarify how current Web Services specifications should be used.

6.1 Specification and Status

WS-I Basic Profile Version 1.0a, Final Specification, August 2003

http://www.ws-i.org/Profiles/Basic/2003-08/BasicProfile-1.0a.htm

6.2 Main Concepts

WS-I provides Web Service Profiles that indicate a set of specifications that can be adopted together in a single application. The profile addresses ambiguities in the specifications to prevent interoperability issues between different implementations. It dictates strong requirements stating that specifications "MUST" or "MUST NOT" have certain details. The profile does not contain optional or loosely defined requirements to avoid discrepancies between implementations. Profiles define testable statements that allow messages to be examined to determine if the implementation is conforming to the stated profile. This is an unobtrusive way of testing with concrete messages instead of having to deal directly with the implementation.

WS-I released the Basic Profile 1.0 (WSBasic) in August 2003. This profile addresses XML Schema 1.0, SOAP 1.1, WSDL 1.1 and UDDI 2.0 specifications. The Basic Profile is the platform for other profiles to be built on. The specifications included in the basic profile are the four basic core components of most Web Services. These specifications are widely supported and are being implemented by vendors now. These specifications are covered in more detail in Section 2 of this document. Vendors can modify their implementations to follow the guidelines and requirements specified in the Basic Profile. This increases the interoperability of those Web Services.

The WSBasic profile provides concrete rules that define profile compliance. The profile addresses each specification included and clarifies ambiguities or loosely defined requirements. An example of this is taken from section 4.1 of the WSBasic profile:

> R1000 When a **MESSAGE** contains a `soap:Fault` element, that element **MUST NOT** have element children other than `faultcode`, `faultstring`, `faultactor` and `detail`.

There is restricting the content of the soap:Fault element to elements explicitly described in the SOAP 1.1 specification. This restriction reduced possible interoperability problems related to the interpretation of the specification. The profile provides examples (XML Messages, WSDL snippets, etc) that clearly indicate the incorrect and correct implementations.

The second profile being developed is the Basic Security Profile. This profile is being created by the WS-I Basic Security Profile Working Group. It builds on the WSBasic profile. The focus is on interoperability issues involving security technologies in the following areas:

- Identification and authentication
- Message integrity and message authentication
- Message confidentiality
- Non-repudiation

The Basic Security Profile is based on the following specifications:

- HTTP over TLS ("HTTPS")
- SOAP attachment security (S/MIME V3 and Cryptographic Message Syntax)
- OASIS Web Services Security V1.0

6.3 Recommendation

Level 2: Emerging

WS-I is attempting exactly what is needed with Web Services. It is not attempting to define new standards, but pick a collection of standards and indicate how they should be used. It goes a step further to provide a test platform to verify that implementations are complying with the standards. The composition of WS-I includes all the required players and the market share needed achieve its stated goals of interoperable Web Services

There are some concerns with WS-I. The initial founders (IBM and Microsoft) did not allow Sun Microsystems into the organization initially. Sun is now a member and modified its Java Web Services Developer Pack (JWSDP) to be WSBasic compliant. There are concerns about the WS-I ability to choose between competing specifications coming out of W3C and Oasis.

Interoperability is a requirement for Web Services. The WS-I is the leading candidate for defining how to achieve interoperability. The reason this is level 2 and is that the testing tools for determining WSBasic profile compliance are not published yet, so no one can claim to be implementing any of the profiles yet. Vendors are working on implementing the WSBasic profile, but there are currently no complete implementations yet.

7 Web Services Choreography and Coordination

This section focuses on the concepts of Web Services *choreography* and *coordination*. According to the W3C Web Services Choreography initiative: "It has become clear that taking the next step in the development of Web Services will require the ability to compose and describe the relationships between lower-level services. Although differing terminology is used in the industry, such as orchestration, collaboration, coordination, conversations, etc., the terms all share a common characteristic of describing linkages and usage patterns between Web Services."

The specifications described in this section all contain choreography and coordination capabilities at some level. This section references the following specifications:

- Web Service Choreography Interface (WSCI), an early submission to the W3C Choreography Working Group
- OASIS Web Services Business Process Execution Language (WS BPEL)
- WS-Transaction/WS-Coordination
- OASIS Web Services Composite Application Framework (WS-CAF)

We begin with a description of the W3C Choreography initiative.

7.1 W3C Web Services Choreography

The W3C Web Services Choreography Working Group was initiated in January 2003 as part of the W3C Web Services Activity. The primary goal of the W3C Web Services Choreography Working Group is to create a common interface and composition language to help address choreography. The Working Group believes that Web Service choreography capabilities are a Critical Success Factor in support of several different top-level goals for the nascent W3C Web Services Architecture.

The Web Services Choreography Working Group published an initial Working Draft of requirements in August 2003. The Working Group is continuing to refine these requirements, and an updated version of this document is anticipated in the upcoming months.

7.1.1 Web Services Choreography Concepts

The description of interactions among Web Services - especially with regard to the exchange of messages, their composition, and the sequences in which they are transmitted and received - is an especially important problem. These interactions may take place among groups of services which, in turn, make up a larger, composite service, or which interact across organizational boundaries in order to obtain and process information. The problems of Web Services choreography are largely focused around message exchange and sequencing these messages in time to the appropriate destinations.

In order to fulfill the needs of the Web Services community, these aspects of Web Services must be developed and standardized in an interoperable manner, taking into account the needs of each individual service as well as those of its collaborators and users. Web Services choreography concerns the interactions of services with their users. These users may be other Web Services, applications or human beings.

7.1.2 W3C Web Services Choreography Interface (WSCI)

The Web Service Choreography Interface (WSCI) is an XML-based interface description language that describes the flow of messages exchanged by a Web Service participating in choreographed interactions with other services. It was submitted to W3C in August 2002 by Sun Microsystems, Intalio, SAP, and BEA Systems.

7.1.2.1 Specification and Status

This section references the following specification:

- WSCI Version 1.0 (W3C Note, August 2002)

7.1.2.2 Main Concepts

Dynamic Interfaces

The Web Service Choreography Interface (WSCI) is an XML-based *interface description language* that describes the flow of messages exchanged by a Web Service participating in choreographed interactions with other services. While mechanisms such as WSDL describe the static interfaces of a Web Service, WSCI describes the *dynamic interface* of the Web Service participating in a given message exchange by means of reusing the operations defined for a static interface. WSCI works in conjunction with the Web Service Description Language (WSDL); it can also work with another service definition language that exhibits the same characteristics as WSDL.

It is important to note that WSCI does not address the definition and the implementation of the internal processes that actually drive a message exchange. Rather, the goal of WSCI is to describe the *observable behavior* of a Web Service by describing the interface between an implementation and the message exchange (collaboration) in which it participates. This is illustrated in the following figure:

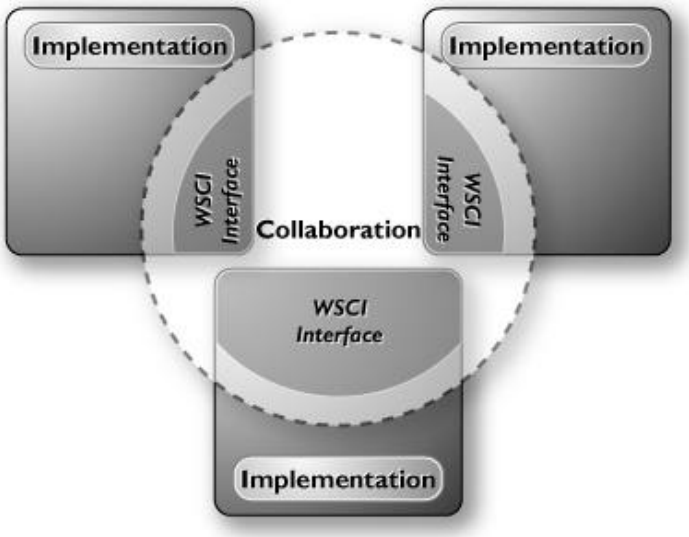

Figure 7.1 WSCI Interfaces and Collaboration

Source: WSCI Version 1.0 Specification

Web Services "Stack" – Where WSCI Fits

A "stack" of layered standards is emerging that aims to ensure semantic and technical interoperability of Web Services. This stack, developed by the W3C, is still in its early stages. Several additional layers are needed in order to enable true Web Service collaborations. Other standards are, in parallel, building semantics and interoperability for business processes and collaborations in a top-down approach. It is anticipated that these two stacks will meet in the middle.

WSCI works on top of the current Web Service stack and below layers in the emerging Web Service architectural model that may be thought of as process or collaboration modeling layers. This is illustrated in the following figure:

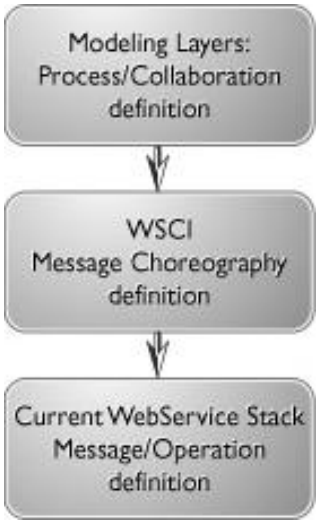

Figure 7.2 WSCI Placement in the Web Services "Stack"

Source: WSCI Version 1.0 Specification

It is important to note that WSCI is not a "workflow description language"; it is envisaged that this role will be covered by some other specification that would properly address the description of collaborative processes. However, WSCI can describe the observable behavior of a Web Service interacting with a workflow; as well, it can describe the observable behavior of a system that implements a workflow (or which behaves as such).

Key Choreography Characteristics

The following are some of the key choreography characteristics that WSCI supports:

- **Message choreography:** A WSCI interface describes the order in which messages can be sent or received in a given message exchange, the rules which govern such ordering, and the boundaries of a message exchange (when it starts and when it ends)
- **Transaction boundaries and compensation:** A WSCI interface describes which operations have transactional ("all or nothing") capabilities

- **Thread management:** A WSCI interface describes if and how a Web Service is capable of managing multiple conversations (based on the same message exchange) with the same partner or with different partners
- **Connectors:** A WSCI interface describes how the operations performed by different Web Services acting in the same message exchange actually link together
- **Operational context:** A WSCI interface describes how the same Web Service behaves in the context of different message exchanges
- **Dynamic participation:** A WSCI interface describes how the identity of the target service is dynamically selected

7.1.2.3 Assessment

Due to the fact that it is not clear that the WSCI specification will be formally accepted by the W3C Web Services Choreography Working Group, we will not provide an assessment of the WSCI specification. Additionally, since the Working Group has not yet produced a specification, we will not reference any specification in particular. We will instead assess the Working Group itself, and will refer to the W3C Web Services Choreography Requirements specification Version 1.0, as this is the only existing specification that the Working Group has produced to this date.

Table 7.1 Assessment of WSCI

Category	Information	Rating
Specification phase	W3C Working Draft	**LOW**
Open standard	YES	**HIGH**
Potential to become open standard		**N/A**
Rate of advancement	Publication date: **August 2003**	**HIGH**
Potential impact on Web Services		**HIGH**
Maturity level of consortium	9 years	**HIGH**
Number of implementations		**N/A**

7.1.2.4 Implementations

N/A

7.1.2.5 Recommendation

Level 2: Emerging

We believe that the work of the W3C Web Services Choreography Working Group is highly important and bears close watching. At this point, the Working Group is still receiving submissions – so it is unclear as to whether the Working Group will adopt WSCI, whether in whole or in part. We foresee the work of this Working Group as having a very high impact on the advancement of Web Services.

7.1.3 OASIS Web Services Business Process Execution Language (WS BPEL)

The OASIS Web Services Business Process Execution Language (WS BPEL) TC was formed in April 2003 to continue the development of the Business Process Execution Language for Web Services (BPEL4WS) specification. The original BPEL4WS specification was authored by IBM, Microsoft, BEA Systems, SAP, and Siebel Systems.

BPEL4WS provides a language for the formal specification of business process behavior based exclusively on Web Services.

7.1.3.1 Specification and Status

This section references the following specification:

- Business Process Execution Language for Web Services (BPEL4WS) Version 1.1 (May 2003)

7.1.3.2 Main Concepts

Business Processes

A BPEL4WS process is a reusable definition that can be deployed in different ways and in different scenarios, while maintaining a uniform application-level behavior across all of them. The following is a simple example of a BPEL4WS process for handling a purchase order:

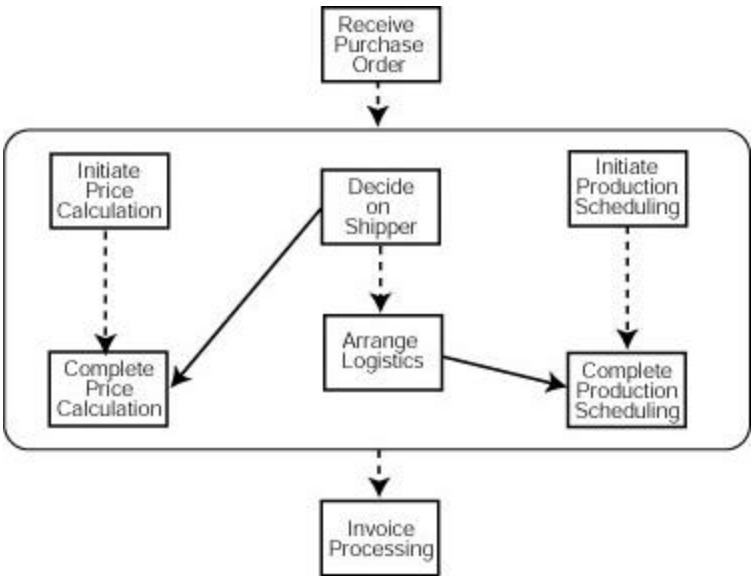

Figure 7.3 Example of BPEL4WS Process

Source: BPEL Version 1.1 Specification

This scenario exemplifies the complex degree to which BPEL4WS can model business processes. On receiving the purchase order from a customer (top), the process initiates three tasks concurrently: calculating the final price for the order (left), selecting a shipper (middle), and scheduling the production and shipment for the order (right). While some of the processing can proceed concurrently, there are control and data dependencies between the three tasks. In particular, the shipping price is required to finalize the price cal-

culation (arrow from "Decide on Shipper" to "Complete Prices Calculation"), and the shipping date is required for the complete fulfillment schedule (arrow from "Arrange Logistics" to "Complete Production Scheduling"). When the three tasks are completed, invoice processing can proceed and the invoice is sent to the customer (bottom).

Partner Links

The interaction with each partner in the business process above occurs through Web Service interfaces, and the structure of the relationship at the interface level is encapsulated in what is called a *partner link*. A partner link is used to directly model peer-to-peer conversational partner relationships. Partner links define the shape of a relationship with a partner by defining the message and port types (from a WSDL definition) used in the interactions in both directions. Partner links also define roles – for example, the roles of "buyer" and "seller" would be applicable to the above scenario.

Transactions and Compensation

BPEL4WS includes transactional capabilities for business processes, through its definition of two types of transactions:

- **Long-Running Transactions (LRTs):** Transactions that span long durations and for which *compensation activities* may be required if reversal of the transaction is necessary
- **ACID (Atomicity/Consistency/Isolation/Durability) transactions:** Also known as *atomic transactions*, ACID transactions are limited to local updates because of trust issues and because locks and isolation cannot be maintained for long periods

A *compensation activity* is an activity associated with a Long Running Transaction that effectively "undoes" the activity of the LRT through a cancellation-type operation. For example, a compensation activity for a purchase order activity would result in the status of the pertinent purchase order being changed to "Cancelled". This differs from an ACID transaction that may involve the transfer of funds from one bank account to another, in which case it is crucial that the transaction as a whole either succeeds or fails completely.

The BPEL4WS specification does not explicitly define the mechanisms for transactional activity, but instead defers to the WS-Transaction specification (discussed later in this section) for this functionality. It is unclear at this time whether or not the WS BPEL specification that emerges from the OASIS activity will maintain this deference, or defer to another specification such WS-CAF (also discussed later in this section).

BPEL4WS and WSCI

Although BPEL4WS and WSCI are sometimes considered as competing specifications, they actually are quite different from each other. While WSCI describes the observable behavior

of a Web Service through description of its interfaces (i.e. its perspective is from the point of view of the Web Service itself), BPEL4WS describes the behavior of a business process based on interactions between the process and its partners. That is, BPEL4WS is actually at a "higher point" in the emerging Web Services stack than WSCI – it is at the "process or collaboration modeling" layer described in the previous WSCI section. At the

core of the BPEL4WS process model is the peer-to-peer interaction between Web Services interfaces defined in WSDL.

Additionally, while WSCI defines all choreography aspects within the context of individual Web Services only and simply connects the interfaces at the global model (process) level, BPEL4WS defines the choreography aspects (e.g. flow of control) at the process level that involves two or more Web Service interfaces. This process level choreography defines which parts of the process execute in parallel, which execute in sequence, conditional flow of control at different parts in the process, exceptions and compensations etc.

Future Releases

The OASIS WS BPEL TC is expecting to release an initial "Editor's Version" of the updated specification in December 2003, with an approved version released in February or March 2004.

7.1.3.3 Assessment

Table 7.2 Assessment of WS BPEL

Category	Information	Rating
Specification phase	OASIS specification in process	**LOW**
Open standard	YES	**HIGH**
Potential to become open standard		**N/A**
Rate of advancement	Publication date: **May 2003**	**HIGH**
Potential impact on Web Services		**HIGH**
Maturity level of consortium	10 years	**HIGH**
Number of implementations	2	**LOW**

7.1.3.4 Implementations

- Collaxa: http://msdn.microsoft.com/webservices/building/wse/default.aspxhttp://www.collaxa.com/product.bpel11.html
- OpenStorm ChoreoServer: http://www.openstorm.com/overview.shtml

7.1.3.5 Recommendation

Level 2: Emerging

We believe that the emerging WS BPEL work is highly important and bears close watching, and we foresee the specification as having a very high impact on the advancement of Web Services, particularly in the area of cross-organization/cross-agency business process interactions.

7.1.4 Web Services Transaction (WS-Transaction)/ Web Services Coordination (WS-Coordination)

The WS-Transaction and WS-Coordination specifications are part of the Global XML Web Services Architecture (GXA). They are discussed together here because of their close dependencies. Both specifications were authored by Microsoft, IBM, and BEA Systems.

WS-Transaction/WS-Coordination together specify the transactional properties of Web Services (WS-Transaction) and the coordination (WS-Coordination) between Web Services, to include the *context* within which Web Services interact.

7.1.4.1 Specification and Status

This section references the following specifications:

- WS-Transaction Version 1.0 (August 2002)
- WS-Coordination Version 1.0 (August 2002)

7.1.4.2 Main Concepts

Coordination Types

WS-Transaction defines two *coordination* types that essentially characterize transactions as either "fine-grained" or "course-grained" transactions. *Atomic transactions* are “all or nothing” transactions that are used to coordinate activities having a short duration and executed within limited trust domains; they are more "fine-grained" in nature. In contrast, *business activities* are used to coordinate activities that are long in duration and that may apply business logic to handle business exceptions; they are more "coarse-grained" in nature. Because of the long duration of business activities, data resources cannot be locked as with atomic transactions—rather, actions are applied immediately and are permanent. A Web Services application can include both atomic transactions and business activities.

Coordination Process

Each activity in a transaction is coordinated by a *coordination service*. Each participant in an activity registers with the coordination service for that activity through a *registration service*. In order to link the various activities participating in a transaction, messages between parties carry a *coordination context*.

The following example illustrates a coordination context supporting a transaction service:

```
[01] <S:Header>
[02]        . . .
[03]   <wscoor:CoordinationContext>
[04]     <wsu:Identifier>http://abc.com</wsu:Identifier>
[05]     <wsu:Expires>2002-08-31T13:20:00-05:00</wsu:Expires>
[06]     <wscoor:CoordinationType>
[07]       http://schemas.xmlsoap.org/ws/2002/08/wstx
[08]     </wscoor:CoordinationType>
[09]     <wscoor:RegistrationService>
[10]       <wsu:Address>
[11]         http://xyzregistrationservice.com
[12]       </wsu:Address>
[13]     </wscoor:RegistrationService>
[14]   </wscoor:CoordinationContext>
```

```
[15]        . . .
[16] </S:Header>
```

In the above example, an atomic transaction is indicated through the value specified by WS-Transaction that denotes an atomic transaction [line 07]. The URI of the registration service with which all Web Services wishing to participate in the transaction register is also specified [line 11].

The WS-Transaction specification is to be split into two specifications, each concentrating on a single transaction type. Atomic transactions are described in a specification known as "WS-AtomicTransaction", which was released in September 2003. Business activities are described in a specification known as "WS-BusinessActivity", whose release is forthcoming.

7.1.4.3 Assessment

Table 7.3 Assessment of WS-Transaction/WS-Coordination

Category	Information	Rating
Specification phase	Initial public draft release	**LOW**
Open standard	NO	**LOW**
Potential to become open standard		**MEDIUM**
Rate of advancement	Publication date: **August 2002**	**LOW**
Potential impact on Web Services		**HIGH**
Maturity level of consortium		**N/A**
Number of implementations	None	**LOW**

7.1.4.4 Implementations

None.

7.1.4.5 Recommendation

Level 3: Questionable

Although we believe that WS-Transaction and WS-Coordination can have a high impact on Web Services, they are not being developed within an open standards consortium. We believe that the capabilities that they specify (the ability to define transactional aspects of Web Services and to coordinate activities among multiple Web Services) are critical for the advancement of Web Services. If these specifications are ever transferred to an open standards consortium, they may be considered as "Level 2: Emerging".

7.1.5 OASIS Web Services Composite Application Framework (WS-CAF)

The Web Services Composite Application Framework (WS-CAF) is a collection of specifications that propose interoperable mechanisms for managing shared context between multiple Web Services acting in combination, and ensuring that business processes achieve predictable results and recovery from failure. The WS-CAF specifications com-

plement other Web Services specifications in the areas of security, reliable messaging, choreography, and transactions.

The areas covered by WS-CAF are similar to those covered by WS-Transaction and WS-Coordination. The specifications are authored by Arjuna Technologies Limited, Fujitsu Software, IONA Technologies PLC, Oracle Corp and Sun Microsystems. A new OASIS TC was formed in September 2003 to continue the development of the WS-CAF specifications.

7.1.5.1 Specification and Status

This section references the following specifications:

- Web Services Context (WS-CTX) Version 1.0 (July 2003)
- Web Services Coordination Framework (WS-CF) Version 1.0 (July 2003)
- Web Services Transaction Management (WS-TXM) Version 1.0 (July 2003)

7.1.5.2 Main Concepts

The Specification Areas

The following is a brief description of each of the three specifications that comprise WS-CAF.

- **Web Service Context (WS-CTX):** A lightweight framework for simple context management. Ensures that all Web Services participating in a composite application share a common context and can exchange information about a common outcome.
- **Web Service Coordination Framework (WS-CF):** A sharable mechanism to manage context augmentation and lifecycle, and guarantee message delivery. Defines a coordinator that provides additional features for persisting context operations and guaranteeing the notification of outcome messages to the participants.
- **Web Services Transaction Management (WS-TXM):** Comprises three distinct protocols for interoperability across multiple transaction managers and supporting multiple transaction models (two phase commit, long running actions, and business process flows). While WS-CF is responsible only for notifying the participants of the outcome, WS-TXM defines a protocol for the participants to coordinate outcomes with each other and make a common decision about how to behave, especially in the case of failure.

WS-CAF specifications are categorized into multiple *domains*, as shown in the following figure:

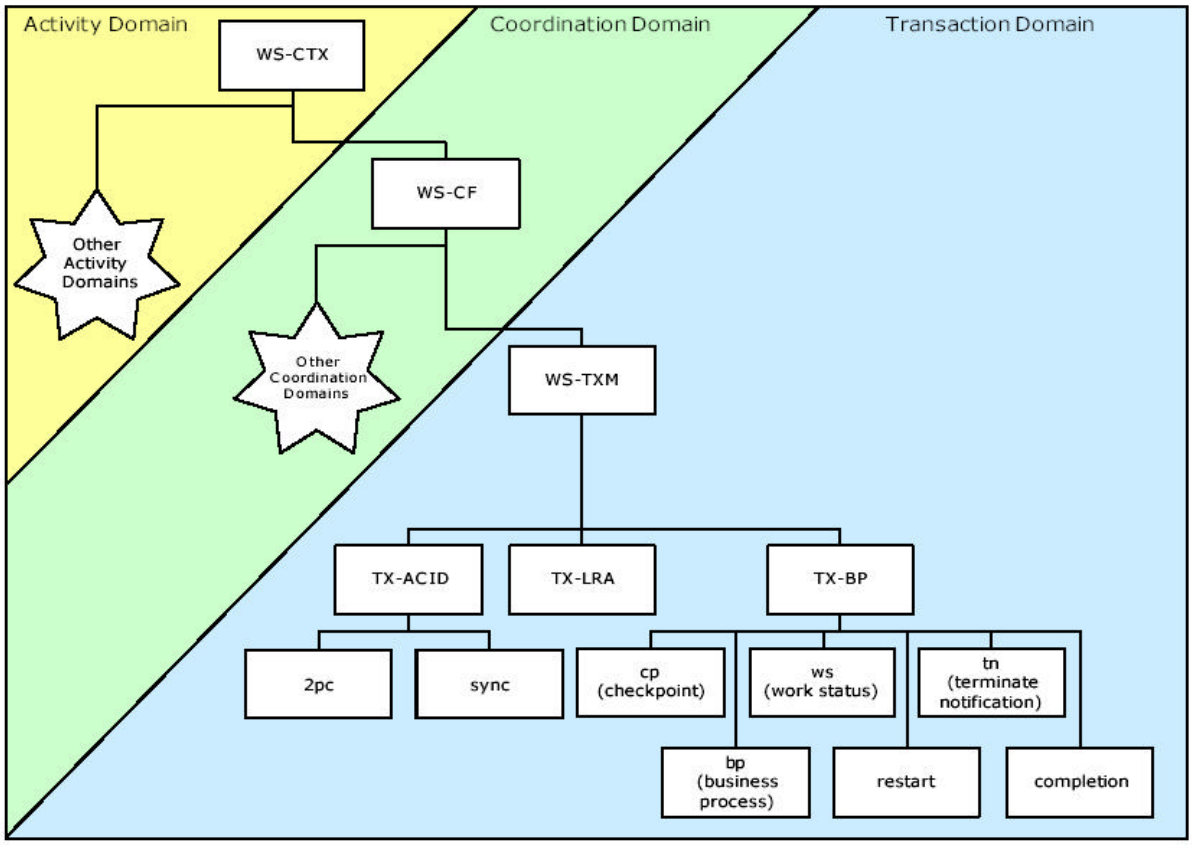

Figure 7.4 WS-CAF Specifications by Domain

Source: WS-CAF Primer

As shown in the above figure, WS-CAF concepts are based on the assumption that multiple Web Services are often placed into various relationships to accomplish a common purpose and therefore at a minimum need a way to share common context *(the Activity Domain)*, and at a maximum need a way to coordinate results *(the Coordination Domain)* into a single, potentially long-running larger unit of work with predictable results despite failure conditions *(the Transaction Domain)*.

WS-CAF and Other Specifications

The WS-CAF specifications overlap in high-level mission with other specifications such as WS-Transaction and WS-Coordination. WS-CAF states that it "can use any transaction protocol in place of or in addition to the neutral protocols defined in WS-TXM", although it remains to be seen how seamless the interoperability between WS-CAF and other transaction protocols might be.

It is also conceivable that BPEL4WS could utilize WS-CAF for its transaction and coordination requirements, although there has been no stated intention to do so.

7.1.5.3 Assessment

Table 7.4 Assessment of WS-CAF

Category	Information	Rating

Category	Information	Rating
Specification phase	OASIS specification in process	**LOW**
Open standard	YES	**HIGH**
Potential to become open standard		**N/A**
Rate of advancement	Publication date: **July 2003**	**HIGH**
Potential impact on Web Services		**HIGH**
Maturity level of consortium	10 years	**HIGH**
Number of implementations	None	**LOW**

7.1.5.4 Implementations

None.

7.1.5.5 Recommendation

Level 2: Emerging

We believe that the emerging WS-CAF work is highly important and bears close watching, and we foresee the specification as having a very high impact on the advancement of Web Services, particularly with the ability to define transactional aspects of Web Services and to coordinate activities among multiple Web Services.

7.2 General Recommendations

The area of choreography and coordination is a relatively new and much-anticipated area for Web Services. We believe that advancements in this area will advance Web Services to new heights that will enable inter-organization and inter-agency collaborations.

The following is a summary of the recommendations given in this section:

- We believe that the work of the W3C Web Services Choreography Working Group is highly important and bears close watching. At this point, the Working Group is still receiving submissions – so it is unclear as to whether the Working Group will adopt WSCI, whether in whole or in part.
- We believe that the emerging WS BPEL work is highly important and bears close watching, and we foresee the specification as having a very high impact on the advancement of Web Services, particularly in the area of cross-organization/cross-agency business process interactions.
- Although we believe that WS-Transaction and WS-Coordination can have a high impact on Web Services, they are not being developed within an open standards consortium. If this occurs, their current Level 3 status may then be considered for upgrade to “Level 2: Emerging”.
- We believe that the emerging WS-CAF work is highly important and bears close watching, and we foresee the specification as having a very high impact on the advancement of Web Services, particularly with the ability to define transactional

aspects of Web Services and to coordinate activities among multiple Web Services.

7.3 References

W3C Web Services Choreography Working Group:
http://www.w3.org/2002/ws/chor/

W3C Web Services Choreography Requirements Version1.0:
http://www.w3.org/TR/2003/WD-ws-chor-reqs-20030812/

WSCI Version 1.0:
http://www.w3.org/TR/wsci/

Business Process Execution Language for Web Services (BPEL4WS) Version 1.1:
http://www-106.ibm.com/developerworks/webservices/library/ws-bpel/

Joseph M. Chiusano, "Web Services Security and More: The Global XML Web Services Architecture (GXA)", Developer.com, March 2003

Prasad Yendluri, "Web Services Choreography", WebServices.org, September 2003

WS-Transaction Version 1.0:
http://msdn.microsoft.com/library/en-us/dnglobspec/html/ws-transaction.asp

WS-Coordination Version 1.0:
http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnglobspec/html/ws-coordination.asp

WS-CAF Primer:
http://developers.sun.com/techtopics/webservices/wscaf/primer.pdf

Web Services Context (WS-CTX) Version 1.0:
http://developers.sun.com/techtopics/webservices/wscaf/wsctx.pdf

Web Services Coordination Framework (WS-CF) Version 1.0:
http://developers.sun.com/techtopics/webservices/wscaf/wscf.pdf

Web Services Transaction Management (WS-TXM) Version 1.0:
http://developers.sun.com/techtopics/webservices/wscaf/wstxm.pdf

8 Web Services and Discovery

This section focuses on the automatic discovery of Web Services. In an earlier section, we discussed the concept a service-oriented architecture (SOA), and the "publish/find/bind" process. In this section we examine the mechanisms behind this process – registries that store and maintain Web Services descriptions.

We examine the two most prominent e-business registry specifications today: Universal Description, Discovery, and Integration (UDDI) and ebXML Registry, and we discuss how ebXML Registry provides functionality beyond that of UDDI to encompass the collaboration phase of e-business. We also provide a high-level comparison between the two registry types.

8.1 Universal Description, Discovery, and Integration (UDDI)

The UDDI project began in October 2000 as a collaboration between Microsoft, Ariba, and IBM. Its main goal was to speed interoperability and adoption for Web Services through the creation of standards-based specifications for service description and discovery, and the shared operation of a business registry on the Web. Before the UDDI project, there was no industry-wide, accepted approach for businesses to reach their customers and partners with information about their products and Web Services. UDDI enables enterprises to quickly and dynamically discover and invoke Web Services, both internally (to the enterprise) and externally.

The initial idea behind UDDI was that software companies, standards bodies, and programmers would populate the public "UDDI Business Registry" with descriptions of different types of services, while businesses would populate the registry with descriptions of the services they support. Marketplaces, search engines, and business applications would then query the registry to discover services at each others' companies. Businesses would also use this data to facilitate easier integration with each other over the Web. UDDI may also be employed as a "private" registry (i.e. behind a firewall) that is hosted by an e-marketplace, a standards body, or a consortium of organizations that participate in a given industry.

8.1.1 Specification and Status

This section references the following specification:

- Universal Description, Discovery, and Integration (UDDI) Version 3.0 (OASIS TC Approved Specification, July 2002)

Originally a vendor-driven specification, UDDI was transferred into OASIS in July 2002.

8.1.2 Main Concepts

UDDI Information Model

The primary focus of the UDDI information model is business information. The UDDI information model consists of the following four "core" data structures:

- businessEntity

- businessService
- bindingTemplate
- tModel

The *businessEntity* data structure is "base structure" of UDDI. A businessEntity describes a business or other organization that typically provides Web Services. It contains descriptive information about the business or provider and the services it offers, such as:

- Names and descriptions in multiple languages
- Contact information
- Classification information

The *businessService* data structure represents a logical grouping of Web Services that a business provides. It should be noted that at this level, there is no technical information provided about these services - rather, this structure allows the ability to assemble a set of services under a common rubric. An example of a businessStructure would be a set of Purchase Order Web Services (submission, confirmation, and notification) that are provided by a business.

The structure that is used to describe the technical information about a Web Service is known as a *bindingTemplate.* Each bindingTemplate structure represents an individual Web Service, and contains either the access point for a given service, or an indirection mechanism that will lead one to the access point.

The core structure in the UDDI information model is a *tModel*, or "technical model". A tModel describes a technical model representing a reusable concept, such as a Web Service type, a protocol used by Web Services, or a category system. An example of a tModel would be a WSDL document that describes a particular Web Service. Each distinct specification, transport, protocol, or namespace within a UDDI registry is represented by a tModel, and tModels are therefore used by multiple bindingTemplates. This allows tModels to be used to promote interoperability between software systems. It should be noted that a UDDI registry does not actually store the content that is denoted by a tModel, but rather references its location.

These structures, and the relationships between them, are represented in the following figure:

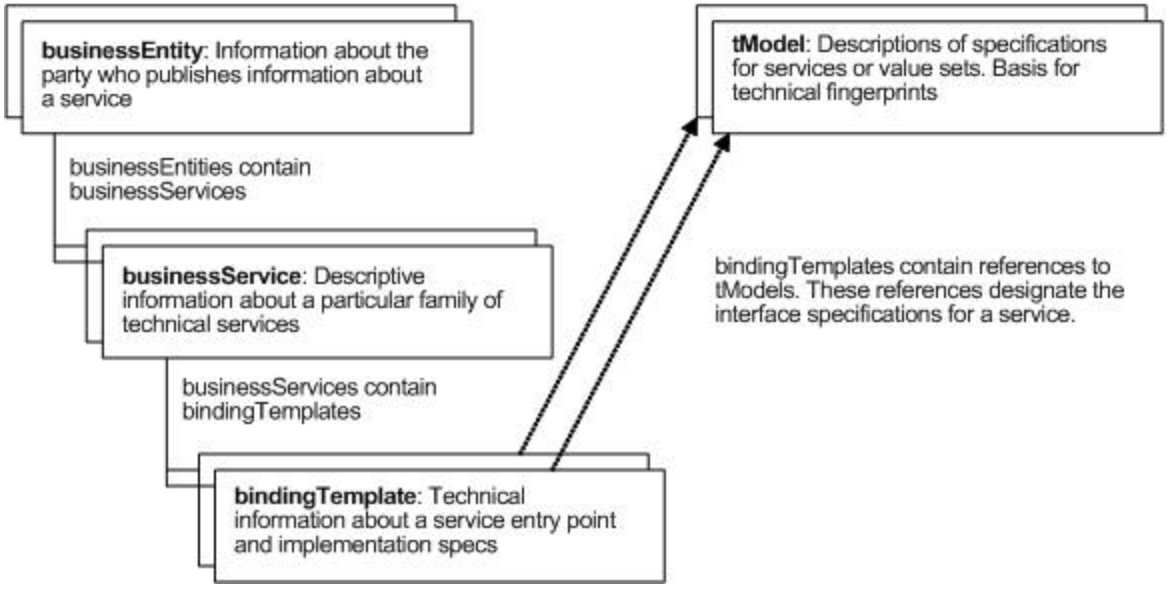

Figure 8.1 UDDI Core Data Structures

Source: UDDI Version 3.0 Specification

Version 3.0 Features

The UDDI Version 3.0 specification contains features that render it quite different from the UDDI Version 2.0 specification. Some of these features are:

- **Multi-Registry Support:** Previous versions of UDDI did not permit the publishing of across multiple registries. This capability is now possible with Version 3.0.
- **Digital Signature Support: Allows UDDI entities to be digitally signed, thereby contributing a higher level of data integrity and authenticity**
- **Policies:** Enables various decisions to be enforced by policies, contributing to more consistent handling of contents
- **Publish/Subscribe:** A new Subscription API includes robust support for synchronous or asynchronous notification of registry events to users

8.1.3 Assessment

Table 8.1 Assessment of UDDI

Category	Information	Rating
Specification phase	OASIS TC Approved Specification	**MEDIUM**
Open standard	YES	**HIGH**
Potential to become open standard		**N/A**
Rate of advancement	Publication date: **July 2002**	**LOW**
Potential impact on Web Services		**HIGH**
Maturity level of consortium	10 years	**HIGH**

Category	Information	Rating
Number of implementations	None	**MEDIUM**

8.1.4 Implementations

UDDI implementations fall into two categories:

- Public
- Private

The number of implementations listed above includes private (COTS) implementations. However, there are current four public UDDI registries operated by the following organizations:

- IBM
- Microsoft
- SAP
- NTT-Com

There are no known implementations that fully support UDDI Version 3.0.

8.1.5 Recommendation

Level 2: Emerging

We believe that UDDI serves a very important purpose for both DISA and the federal government, as usage of Web Services increases, and its adoption is currently on the rise. The mechanisms that UDDI provide for both management and discovery of Web Services descriptions will serve to advance adoption of Web Services through increased discovery capabilities. Because the UDDI Version 3.0 specification is still under development, we recommend that DISA consider using UDDI Version 2.0 specification implementations for the time being in order to advance its Web Services efforts, and upgrade to Version 3.0 when it becomes an OASIS standard and when an acceptable number of implementations are available.

8.2 ebXML Registry

The ebXML Registry specification was created as part of the 18-month ebXML initiative that ended in May 2001, after which time it was moved into OASIS. An ebXML Registry provides a mechanism by which XML and non-XML artifacts (including Web Services descriptions) can be stored, maintained, and automatically discovered, thereby increasing efficiency in XML-related development efforts.

8.2.1 Specification and Status

This section references the following specification:

- OASIS/ebXML Registry Information Model (RIM) Version 2.5 (OASIS TC Approved Specification, June 2003)
- OASIS/ebXML Registry Services (RS) Specification Version 2.5 (OASIS TC Approved Specification, June 2003)

The OASIS/ebXML Registry TC plans to release these specifications for OASIS review and vote in late 2003 or early 2004.

8.2.2 Main Concepts

ebXML Registry Information Model

Unlike UDDI whose primary focus is business information, the main focus of the ebXML Registry information model is more general to encompass XML and non-XML artifacts. Therefore, the ebXML Registry information model is more abstract in nature than that of UDDI.

The ebXML Registry information model consists of two "core" data structures (known as *classes):*

- RegistryObject
- RegistryEntry

A *RegistryObject* provides metadata for a stored *RepositoryItem* (the term used to refer to that actual object that is stored) – such as name, object type, identifier, description, etc. A RegistryObject can represent many different types of RepositoryItems, from XML schemas, to classification schemes, to Web Service definitions.

In contract, a *RegistryEntry* is used to represent "catalog-type" metadata about RepositoryItems – that is, metadata about the current state of a RepositoryItem in the registry (e.g. version, status, stability). Consequently, the metadata associated with a RegistryEntry is (in general) more "fluid" than that associated with a RegistryObject. The RegistryEntry class inherits form the RegistryObject class.

Version 2.5 Features

As with UDDI, the OASIS/ebXML Registry Version 2.5 specifications contain features that render them quite different from the OASIS/ebXML Registry Version 2.0 specifications. Some of these features are:

- **Cooperating Registries:** Similar in concept to UDDI's multi-registry support
- **Event Notification:** Similar in concept to UDDI's publish/subscribe feature
- **Content Management Services:** Provide content validation and cataloging capabilities
- **OASIS XACML Support:** Allows fine-grained access control policies to be defined for ebXML Registry

ebXML Registry and UDDI

ebXML Registry contains classes for representing Web Services that are generally equivalent to the four "core" data structures of UDDI. The OASIS/ebXML Registry TC has released a Technical Note called "Registering Web Services in an ebXML Registry" that describes both the representation of Web Services in an ebXML Registry and the process for registering them.

A large distinction between ebXML Registry and UDDI in addition to their primary focuses and information models is the general "phases" of e-business with which each reg-

istry type is associated. There are two general ways in which an e-business registry (such as ebXML Registry or UDDI) may be used: for *discovery* and for *collaboration*. Both UDDI and ebXML Registry allow for discovery of businesses, their Web Services, and the technical interfaces they make available. However, UDDI is focused exclusively on this discovery aspect, while ebXML Registry is focused on both discovery and collaboration. The primary focus of ebXML Registry extends beyond that of UDDI into collaboration. Due to its focus on storing and maintaining XML artifacts, an ebXML registry can enable both collaborative development of XML artifacts within an organization and runtime collaboration between trading partners. For example, users can create XML artifacts and submit them to an ebXML registry for use and potential enhancement by other users.

We believe that ebXML Registry and UDDI will co-exist and continue to be utilized in their areas of strength – ebXML Registry for discovery and collaboration, and UDDI for discovery. This view is further explained in the WebServices.org article titled "UDDI and ebXML Registry: A Co-Existence Paradigm". We also foresee greater interoperability between the two registry types as described in the ebXML Forum article "UDDI and ebXML Registry: A Three-Tier Vision".

8.2.3 Assessment

Table 8.2 Assessment of ebXML Registry

Category	Information	Rating
Specification phase	OASIS TC Approved Specification	**MEDIUM**
Open standard	YES	**HIGH**
Potential to become open standard		**N/A**
Rate of advancement	Publication date: **June 2003**	**HIGH**
Potential impact on Web Services		**HIGH**
Maturity level of consortium	10 years	**HIGH**
Number of implementations	None	**LOW**

8.2.4 Implementations

While a number of vendor implementations of OASIS/ebXML Registry 2.0 exist, no vendor implementations of Version 2.5 have been identified.

8.2.5 Recommendation

Level 2: Emerging

We believe that the current adoption of ebXML Registry (in general) has been very low. We attribute this mostly to a general perception that an XML registry is not necessary, or is a "nice to have". As adoption of XML grows both within DISA and the federal government—particularly the creation of XML schemas—we foresee the need for an XML registry such as ebXML Registry increasing.

For the present time, however, we recommend that DISA hold off on adopting ebXML Registry until the Version 2.5 specifications become OASIS standard and reach a Version 3.0 status. After that time, an assessment should be made regarding available implementations, and further consideration should be given to implementing ebXML Registry.

8.3 General Recommendations

The utilization of service-oriented architectures (SOAs) calls for efficient mechanisms by which to discover Web Services descriptions, such as WSDL documents.

The following is a summary of the recommendations given in this section:

- We believe that UDDI serves a very important purpose for both DISA and the federal government, as usage of Web Services increases, and its adoption is currently on the rise. Because the UDDI Version 3.0 specification is still under development, we recommend that DISA consider using UDDI Version 2.0 specification implementations for the time being in order to advance its Web Services efforts, and upgrade to Version 3.0 when it becomes an OASIS standard and when an acceptable number of implementations are available.
- We believe that the current adoption of ebXML Registry (in general) has been very low. However, as adoption of XML grows both within DISA and the federal government—particularly the creation of XML schemas—we foresee the need for an XML registry such as ebXML Registry increasing. Once the OASIS/ebXML Registry Version 2.5 specifications reach a Version 3.0 status, an assessment should be made regarding available implementations and further consideration should be given to implementing ebXML Registry.
- We believe that ebXML Registry and UDDI will co-exist and continue to be utilized in their areas of strength – ebXML Registry for discovery and collaboration, and UDDI for discovery.

8.4 References

Joseph M. Chiusano, “UDDI and ebXML Registry: A Co-Existence Paradigm”, WebServices.org, April 2003

Joseph M. Chiusano, “UDDI and ebXML Registry: A Three-Tier Vision”, ebXML Forum, August 2003

Universal Description, Discovery, and Integration (UDDI) Version 3.0:
http://uddi.org/pubs/uddi_v3.htm

UDDI Version 3.0 Features List:
http://uddi.org/pubs/uddi_v3_features.htm

Registering Web Services in an ebXML Registry:
http://xml.coverpages.org/xmlPapers200305.html#WS-ebXML

OASIS/ebXML Registry Information Model (RIM) Version 2.5:
http://www.oasis-open.org/committees/regrep/documents/2.5/specs/ebrim-2.5.pdf

OASIS/ebXML Registry Services (RS) Specification Version 2.5:
http://www.oasis-open.org/committees/regrep/documents/2.5/specs/ebrs-2.5.pdf

9 The Semantic Web

> "*The Semantic Web is an extension of the current web in which information is given well-defined meaning, better enabling computers and people to work in cooperation.*" *-- Tim Berners-Lee, James Hendler, Ora Lassila, The Semantic Web (*http://www.scientificamerican.com/article.cfm?articleID=00048144-10D2-1C70-84A9809EC588EF21&catID=2*), Scientific American, May 2001*

The W3C has launched an activity known as the Semantic Web, with the goal of making resources on the Internet accessible to automated tools, rather than limiting the use of most of the information to human readers (e.g., via a web browser). This is a very broad concept, and the activity is broad in scope, but the W3C intends to be a facilitator for the creation of standards relevant to attaining their goals.

The W3C Semantic Web activity primarily includes the Resource Description Framework (RDF) Core and Web Ontology working groups. RDF is a set of layered specifications into which the principal technologies of the Semantic Web fit. This section discusses W3C standards activities related to the Semantic Web and related efforts.

9.1 W3C Web Ontology Language (OWL)

The Web Ontology Language (OWL) is a revision of the DAML+OIL web ontology language that builds on the development and use of DAML+OIL. OWL is intended to be used when the information contained in documents needs to be processed by applications, rather than presented. OWL can be used to explicitly represent the meaning of terms in vocabularies, their properties, and the relationships between them.

OWL provides three increasingly expressive (in the logical sense) sublanguages, each designed for use by specific sub-communities of implementers and users.

OWL Lite, the least expressive of the sublanguages, supports those users primarily needing a classification hierarchy and simple constraints. OWL Lite provides a quick migration path for thesauri and other taxonomies. Owl Lite also has a lower formal complexity than OWL DL, which is the next more expressive of the sublanguages.

OWL DL supports those users who want the Description Logic properties of maximum expressiveness while retaining computational completeness (all conclusions are guaranteed to be computed), and of decidability (all computations will finish in finite time). OWL DL includes all OWL language constructs, but they can be used only under certain restrictions (for example, while a class may be a subclass of many classes, a class cannot be an instance of another class).

OWL Full is meant for users who want maximum expressiveness with no computational guarantees. For example, in OWL Full a class can be treated simultaneously as a collection of individuals and as an individual in its own right. Experience in the development of automated reasoning systems suggests that is unlikely that any reasoning software will be able to support complete reasoning for every feature of OWL Full.

9.1.1 Specification and Status

OWL Web Ontology Language Semantics and Abstract Syntax (W3C Candidate Recommendation 18 August 2003)

http://www.w3.org/TR/owl-semantics/

9.1.2 Main Concepts

Ontology

The OWL Web Ontology Language Use Cases and Requirements document (http://www.w3.org/TR/webont-req/) gives the following description of an ontology:

An ontology defines the terms used to describe and represent an area of knowledge. Ontologies are used by people, databases, and applications that need to share domain information (a domain is just a specific subject area or area of knowledge, like medicine, tool manufacturing, real estate, automobile repair, financial management, etc.). Ontologies include computer-usable definitions of basic concepts in the domain and the relationships among them (note that here and throughout this document, definition is not used in the technical sense understood by logicians). They encode knowledge in a domain and also knowledge that spans domains. In this way, they make that knowledge reusable.

The word ontology has been used to describe artifacts with different degrees of structure. These range from simple taxonomies (such as the Yahoo hierarchy), to metadata schemes (such as the Dublin Core), to logical theories. The Semantic Web needs ontologies with a significant degree of structure. These need to specify descriptions for the following kinds of concepts:

- Classes (general things) in the many domains of interest
- The relationships that can exist among things
- The properties (or attributes) those things may have

Ontologies are usually expressed in a logic-based language, so that detailed, accurate, consistent, sound, and meaningful distinctions can be made among the classes, properties, and relations. Some ontology tools can perform automated reasoning using the ontologies, and thus provide advanced services to intelligent applications such as: conceptual/semantic search and retrieval, software agents, decision support, speech and natural language understanding, knowledge management, intelligent databases, and electronic commerce.

9.1.3 Implementations

As of 2003-08-18 the Owl Implementations page (http://www.w3.org/2001/sw/WebOnt/impls) listed more than a dozen implementations.

9.1.4 Recommendation

Level 2: Ready for early implementers.

Experience with existing implementations shows that they can be used to produce Ontologies that are interoperable.

9.2 DARPA Agent Markup Language – Semantic (DAML-S)

DAML-based Web Service Ontology (DAML-S) – which is set to become the OWL-based Web Service Ontology (OWL-S) – is a framework for description of Web Services at a level of detail sufficient to permit reasoning about services. The usefulness of reasoning has been shown in the area of automated planning, where planning software (the "planner") is capable of generating schedules for the execution of services that, when executed, fulfill the goals and objectives that were input to the planner. Planning, particularly continuous planning or planning under uncertainty, requires being able to track the execution of services; this in turn typically depends on knowledge of the way in which services do what they do. It is this level of description that DAML-S (and OWL-S) targets.

9.2.1 Specification and Status

DAML-S (and OWL-S) 0.9 Draft Release 2003-05

http://www.daml.org/services/daml-s/0.9/

9.2.2 Main Concepts

Service

The class Service represents the highest-level concept in the service ontology.

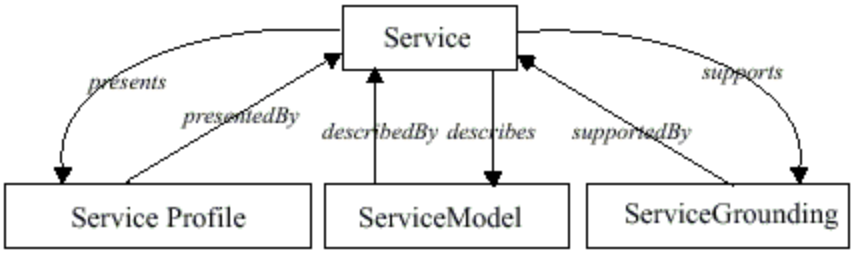

Figure 9.1 The Top Levels of the Service Ontology

Figure 9.1 shows the division of the top of the service ontology into three areas: the Service Profile, the Service Model, and the Service Grounding. The Service Profile describes three types of information for a service: the organization that provides the service, the function that the service computes, and characteristics of the service, such as its parameters, quality of service provided, and so on. The Service Model describes what the service does; a Process Model is a subclass of Service Model that describes how a service does what it does. The Service Grounding describes how to bind to a service, in a way that is very close to WSDL.

Process

A key concept in OWL-S is the process; a process is an activity carried out by an agent, which is typically a Web Service or a client of a Web Service. Processes can be atomic,

simple, or various sorts of composite, distinguished by their control constructs (conditional, choice, parallel, loop, etc.).

Every process can have input and output parameters; input and output parameters may have optional types, which are the OWL classes that they belong to. Processes can also have preconditions, which must be true before the process can be started. If the precondition is not true when the process begins, then some sort of failure occurs. Processes can also have effects, which can be conditional or unconditional.

9.2.3 Implementations

Several service composition systems using DAML-S and various existing planning systems have been implemented. A number of these systems were discussed in the ICAPS 2003 Workshop on Planning for Web Services (http://www.isi.edu/info-agents/workshops/icaps2003-p4ws/program.html).

9.2.4 Recommendation

Level 3: In the research stage, but certainly bears watching.

Recognized world experts are working on this project.

9.3 Topic Maps

The Topic Maps formalism provides a linking technology for working with "information objects", such as texts, electronic documents, or knowledge bases. Topic Map technology can be applied to produce navigational tools such as indexes, cross-references, citation systems, or glossaries without having to modify the target information objects. The Topic Maps formalism supports connecting links together in order to create thesaurus-like interfaces to information objects. There is also support for filtering access to information objects, based on user profiles or on security considerations.

9.3.1 Specification and Status

ISO/IEC 13250:2000 Topic Maps (1999-12-03)

http://www.iso-standards-international.com/iso-13250.htm

XML Topic Maps (XTM) 1.0 Version 1.16 (2001/08/06)

http://www.topicmaps.org/xtm/1.0/xtm1-20010806.html

Published Subjects: Introduction and Basic Requirements (OASIS Published Subjects Technical Committee Recommendation, 2003-06-24)

http://www.oasis-open.org/committees/documents.php?wg_abbrev=tm-pubsubj

9.3.2 Main Concepts

Topics

Everyday language typically talks about topics in relation to texts ('text' being a generic term for a coherent piece of speech or writing). Phrases such as "that question is off topic" suggest that in everyday usage, the connection between a topic and its related

piece of language can be rather loose, perhaps even looser than the connection between language and meaning. In the Topic Maps formalism this everyday notion is generalized, so that topics can be related to any sort of "information object", and not just texts.

In everyday usage the terms 'topic' and 'subject' are interchangeable in many contexts. (In a strict linguistic sense, however, they denote different things. The subject of a sentence or of a proposition is a determinate grammatical or logic entity, the topic is some much vaguer thing which is dependent on context and intent.) The Topic Maps formalism systematically uses the term 'subject' for what is called a topic, in everyday language. The term 'topic' is then reserved for a formal object in the Topic Maps architecture, a formal object that refers to, or indicates, (in an undefined way) a subject.

Every topic indicates some subject. Since the Topic Maps formalism objectifies topics, it is often said that the topic "reifies" the subject — or makes the subject "real" for a Topic Maps system. The creation of a topic that reifies a subject enables a Topic Maps system to manipulate, process, and assign characteristics to the subject by manipulating, processing, and assigning characteristics to the topic that reifies it. When an address for the subject is needed, the address of a topic that reifies it – which acts as its surrogate within the system – can be given.

Associations

Associations express relationships among topics. Topic Maps applications define the nature of such relationships and of the roles played by topics in those relationships.

In logic, relationships are usually considered as being directed: a relationship has a domain and a range, or a subject and an object. Within the topic maps formalism the associations that express relationships are considered to be inherently multidirectional. Instead of directionality, associations use roles to distinguish between the various forms of involvement members have in them.

Occurrences

Occurrences are the "addresses" of topics (or better, of subjects") in information objects; the occurrence may be the "address" of the entire information object, or some range or position within it.

A **topic map** is then a set of topics with their occurrences, together with all of the associations between those topics.

Topics can also occur as published subjects, which are maintained in subject repositories, by well-known publishers.

9.3.3 Implementations

The original Topic Maps architecture was dependent on the HyTime architecture. Since that time a version based on XLink has been developed. There are several implementations currently available. Topic Maps are being used as an output format for data mining tools.

9.3.4 Recommendation

Level 2: Ready for early implementers.

Topic Maps is an ISO specification. However, despite commercial promotion, it has not yet been widely adopted, in the Web Services community. The Topic Maps formalism is consistent with the Semantic Web vision of annotated web pages, but the connection with Web Services is unclear.

10 Web Services Monitoring and Management

The concept of monitoring and managing services or systems is not new. The requirement to do so has always existed in IT departments. Systems must have specific hooks exposed for management software to leverage. Modifications are required when new components or resources are added to the system. The distributed, loosely coupled nature of Web Services provides both a challenge and opportunity to the classic management problem.

Management will be a required to successfully operate large distributed, complex systems built on Web Services. The W3C Web Service Architecture recognizes this fact and has addressed the management requirements. In addition the OASIS Web Services Distributed Management (WSDM) (http://www.oasis-open.org/committees/wsdm) Technical committee in the process of defining the specifications for the management of Web Services and management using Web Services.

10.1 Specification and Status

OASIS Web Services Distributed Management (WSDM) V1.0 Specification, which has a delivery date of Jan 2004.

http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsdm

10.2 Main Concepts

The W3C Web Service Architecture (WSA) specification claims that WSA implementations must be manageable, that Web Services instances must be manageable, and further defines the types of management that must be supported. The specification starts by dictating that a set of standard metrics be used by implementations. It goes on to require a base set of management operations including, but not limited to, configuration control and lifecycle control. It requires a set management events be issued by the WSA to allow monitoring of the system. It specifies that implementations must have a standard methodology for accessing the management capabilities. This allows management capabilities to be interoperable.

The WSA specification then addresses how individual services are to be managed. It indicates that Web Services should expose metrics, configuration, operations and events. The management capabilities should be published so that they can be discovered. All Web Service instances should conform to a standard methodology for accessing the management operations.

Web Services management falls into two areas Management Using Web Services (MUWS) and Management of Web Services (MOWS). The goal is to use Web Services to manage Web Services. This satisfies many of the requirements set forth by the W3C WSA. The management definitions will be defined by WSDL or GWSDL documents. This addresses the interoperability, standard access methodology and discovery of management capabilities.

The OASIS WSDM TC is drafting a specification that defines how the requirements presented in the WSA should be implemented. The final draft is not due until January 2004.

The WSDM TC membership includes the likes of Hewlett-Packard, Computer Associates, IBM, BEA, Sun and others. Two different papers have been submitted to the WSDM, the Web Services Management Framework (WSMF) (http://devresource.hp.com/drc/specifications/wsmf/WSMF-WSM.jsp)

submitted by HP and the Web Services Manageability (WS-Manageability) (ftp://www6.software.ibm.com/software/developer/library/ws-manage.pdf) submitted by IBM, CA and Talking Blocks. There are a lot of similarities between the two papers and all the authors are members of the technical committee. The committee is currently reviewing both papers and creating the draft for the final specification. It appears much of the WS-Manageability paper will be leveraged in the final specification.

10.3 Recommendation

Level 3: Questionable

Web Services will require management but at this time there is no official standard yet. There are current products and solutions for addressing management but there is no clear-cut market leader. These products provide proprietary solutions. Once the draft is finalized many vendors will provide implementations of the specifications. It is important to reevaluate this area after the draft has been released in January 2004.

11 Applications of Web Services

This section includes a sampling of specifications and standards that extend from the core Web Services specifications, and are particularly relevant to DoD C2 systems. These specifications are application specific, but demonstrate how the Web Services framework is being leveraged as a foundation for creating stronger interoperability solutions in some key areas of distributed computing.

11.1 OASIS Web Services for Remote Portlets

Portals provide personalized access to information, applications, processes and people. Typically, portals get information from local or remote data sources, e.g. from databases, transaction systems, syndicated content providers, or remote web sites. They render and aggregate this information into composite pages to provide information to users in a compact and easily consumed form. In addition to pure information, many portals also include applications like e-mail, calendar, organizers, banking, bill presentment, host integration, etc.

OASIS Web Services for Remote Portlets (WSRP) aims to simplify integration of content with portals through a standard set of Web Service interfaces that allow integrating applications to quickly exploit new Web Services as they become available. The specification discussed in this section was jointly developed by the OASIS WSRP and WSIA (Web Services for Interactive Applications) Technical Committees.

11.1.1 Specification and Status

This section references the following specification:

- Web Services for Remote Portlets (WSRP) Version 1.0 (OASIS Standard, August 2003)
  http://www.oasis-open.org/committees/download.php/3343/oasis-200304-wsrp-specification-1.0.pdf

11.1.2 Main Concepts

Portals: The Integration Challenge

Portals and other Web applications render and aggregate information from different sources and provide it in a compact and easily consumable form to an end-user. Among the typical sources of information are Web Services. Traditional data-oriented Web Services, however, require aggregating applications to provide specific presentation logic for each of these Web Services. Furthermore, each aggregating application communicates with each Web Service via its unique interface. This approach is not well suited to dynamic integration of business applications and content as a plug-and-play solution.

The WSRP specification solves this problem by introducing a presentation-oriented Web Service interface that allows the inclusion of and interaction with content from a Web Service. It provides a common protocol and a set of interfaces for presentation-oriented Web Services, allowing aggregating applications to easily “adopt” these Web Services by utilizing generic proxy code.

Portlets

Portlets are presentation-oriented, interactive web application components that are aggregated and displayed by a portal. Portal content is often provided by external services and displayed by specific local portlets running on the portal. The WSRP mechanism for aggregating portlets using proxies is shown in the following figure:

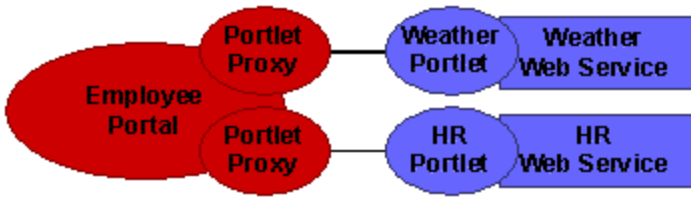

Figure 11.1 WSRP Mechanism for Aggregating Portlets using Proxies

Source: WSRP Version 1.0 Specification

In the above figure, two Web Services that provide HR and weather information are aggregated at an Employee portal using proxies.

11.1.3 Assessment

Table 11.1 Assessment of WSRP

Category	Information	Rating
Specification phase	OASIS Standard	**HIGH**
Open standard	YES	**HIGH**
Potential to become open standard		**N/A**
Rate of advancement		**N/A**
Potential impact on Web Services		**HIGH**
Maturity level of consortium	10 years	**HIGH**
Number of implementations	8	**MEDIUM**

11.1.4 Implementations

The following are three examples of identified implementations:

- Plumtree: http://xml.coverpages.org/PlumtreePortlet.html
- BEA:http://www.bea.com/content/news_events/white_papers/BEA_WL_platform7_ds.pdf
- Sun Microsystems: http://www.sun.com/smi/Press/sunflash/2002-03/sunflash.20020327.4.html

11.1.5 Recommendation

Level 2: Emerging

We believe that WSRP will have a large impact on Web Services through its ability to seamlessly deliver aggregated content to a centralized location. There has also been a strong emphasis in the federal government on portal technology in recent years, which makes WSRP even more attractive. However, although a fair number of WSRP implementations are available, we believe that the freshness of this standard warrants a waiting period before use.

There has also been a strong emphasis in the federal government on portal technology in recent years, which has yielded a high need for open standards in the area of portals and Web Services. Although currently the only prominent open standard in this area, we believe that WSRP will have a large impact on Web Services through its ability to seamlessly deliver aggregated content to a centralized location. Although a fair number of WSRP implementations are available, we believe that the freshness of this standard warrants a waiting period before use.

11.1.6 References

"Enabling Interactive, Presentation-Oriented Content Services Through the WSRP Standard":
http://www.oasis-open.org/committees/download.php/3657/WSRP_paper.html

11.2 Geospatial Web Services

The OpenGIS Consortium (OGC) has actively promoted the development and prototyping of specifications for interoperable Web Services that can share and process geospatial information, at least since 1997, when the first Web Mapping testbed took place. Since then, they have moved into the areas of sensors and sensor Web Services, and into location-based Web Services. In this section we only consider OGC Web Services that are at least far enough along in the specification process to be publicly available. For this to have happened a significant number of different implementations have to have been shown to interoperate.

11.2.1 Specification and Status

OpenGIS Reference Model - The OpenGIS Reference Model (ORM) provides an architecture framework for the ongoing work of the OGC. Further, the ORM provides a framework for the OGC Technical Baseline. The OGC Technical Baseline consists of the currently approved OpenGIS Specifications as well as for a number of candidate specifications that are currently in progress. http://www.opengis.org/info/orm/03-040.pdf

OpenGIS Web Map Service (WMS) Implementation Specification - A Web Map Service produces maps of geo-referenced data. A map is a visual representation of geodata; a map is not the data itself. This specification defines three WMS operations: GetCapabilities returns service-level metadata, which is a description of the service's information content and acceptable request parameters; GetMap returns a map image whose geospatial and dimensional parameters are well-defined; GetFeatureInfo (optional) returns information about particular features shown on a map.
http://www.opengis.org/techno/specs/01-068r3.pdf

OpenGIS Geography Markup Language (GML) Implementation Specification - Geography Markup Language is an XML grammar written in XML Schema for the modeling, transport, and storage of geographic information.
http://www.opengis.org/techno/documents/02-023r4.pdf

OpenGIS Web Feature Service Implementation Specification - The OGC Web Feature Service allows a client to retrieve geospatial data encoded in Geography Markup Language (GML) from multiple Web Feature Services.
http://www.opengis.org/techno/specs/02-058.pdf

OpenGIS Filter Encoding Implementation Specification - This specification defines an XML encoding for filter expressions based on definitions from the OpenGIS Common Catalog Query Language as described in the OpenGIS Catalog Interface Implementation Specification, Version 1.0. http://www.opengis.org/techno/specs/02-059.pdf

OpenGIS Styled Layer Descriptor Implementation Specification - This specification addresses the need for geospatial consumers (either humans or machines) to control the visual portrayal of data. It can be used to portray the output of Web Map Servers, Web Feature Servers and Web Coverage Servers. http://www.opengis.org/techno/specs/02-070.pdf

Web Map Context Documents - The present Context specification states how a specific grouping of one or more maps from one or more map servers can be described in a portable, platform-independent format for storage in a repository or for transmission between clients. A Context document includes information about the server(s) providing layer(s) in the overall map, the bounding box and map projection shared by all the maps, sufficient operational metadata for Client software to reproduce the map, and ancillary metadata used to annotate or describe the maps and their provenance for the benefit of human viewers. A Context document is structured using XML.
http://www.opengis.org/techno/specs/03-036r2.pdf

11.2.2 Main Concepts

Abstract Specifications

All of the specifications or candidate specifications listed above are derived from abstract services definitions, which are specialized to use Web Services as their distributed computing platform. As an organization OGC spent many years first developing a large body of abstract specifications for data and services in the geospatial arena. These specifications cover all the fundamental concepts and technologies of the geospatial industry, and use UML as a description language.

Interoperability

The geospatial industry (principally GIS, mapping, and remote sensing) has long had the goal of providing interoperable services for sharing and processing of geospatial information. A number of government agencies are responsible for disseminating mapping information in a timely manner. Traditional paper maps do not always fulfill this goal. NASA and the commercial remote sensing companies collect huge volumes of geospatial information on a daily basis, which needs to be disseminated and integrated with other information. Location-based information is another component of the equation. Further-

more, traditionally, different vendors have specialized in geospatial data storage, and in geospatial information display. All of this has placed a huge demand on the geospatial industry to achieve interoperability.

11.2.3 Implementations

A large and growing number of implementations are available.

11.2.4 Recommendation

Level 1: Ready for use.

Business, government, and military organizations, principally in Japan, Australia, the EU, Canada, and the US, have adopted OGC standards as the preferred solution for interoperable geospatial data sharing and processing.

11.3 Sensor Web Services

The Sensor Model Language (SensorML) is a set of XML Schemas, which define sensor descriptions in XML. A sensor description describes the characteristics that are required for processing, geo-registering, and assessing the quality of measurements from sensor systems. SensorML schemas work together with the Observations and Measurement schemas, which are defined in a separate specification.

11.3.1 Specification and Status

Sensor Model Language (SensorML) for In-situ and Remote Sensors Specification Version 0.7 (OGC Discussion Paper 2002-12-20)

http://www.opengis.org/techno/discussions/02-026r4.pdf

Observations and Measurements Version 0.9.2 (OGC Recommendation 2003-02-04)

http://www.opengis.org/techno/discussions/03-022r3.pdf

11.3.2 Main Concepts

Observables

Observables are properties of physical entities and phenomena that are capable of being measured and quantified. Each of these can be classified as an Observable type and can be referenced in an Observables dictionary. Observable type definitions include, for example, properties such as temperature, count, rock type, chemical concentration, or radiation emissivity.

Sensors

Sensors are systems that are capable of observing and measuring particular properties. Either by design or as a result of operational conditions, these sensors have particular response characteristics that can be used to determine the values of certain measurements, as well as assess the quality of these measurements. In addition to the response characteristics, the sensor system has properties of location and orientation that allow one to associate the measured values with a particular geospatial location at a particular time.

Observations and Measurements

O&M defines an observation to be an event with a result which is a value describing some phenomenon. An observation feature binds the result to the (spatiotemporal) location where it was made. An observation involves a procedure to determine the value, which may involve a sensor or observer, analytical procedure, simulation or other numerical process. An observation is modeled as a feature within the context of the OGC feature model.

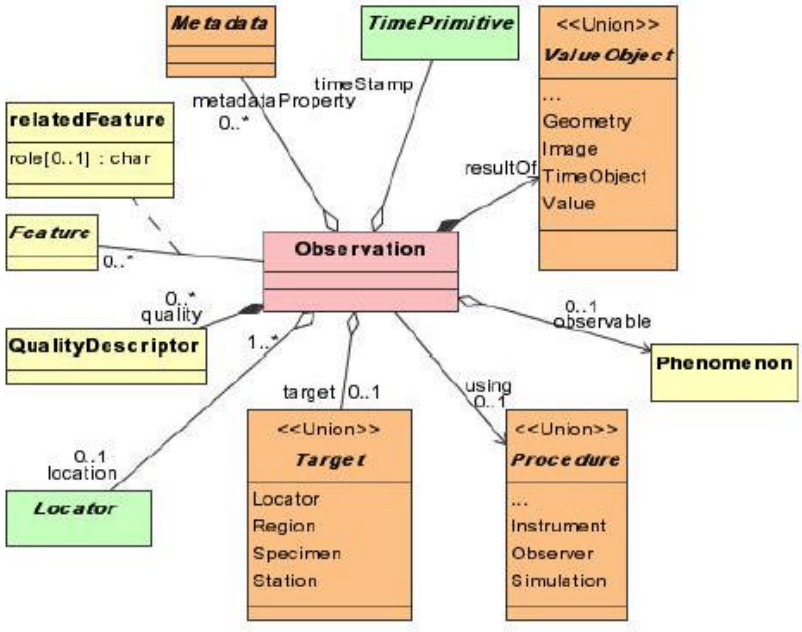

Figure 11.2 O&M Observation Object Model

An observation results in an estimate of the value of a property or phenomenon related to the target of the observation. Values are of various data types, including the primitive types category, quantity, count and boolean, time, location and geometry. The value normally requires a reference system to provide the context for its interpretation and valid operations on it. Common reference systems are the unit of measure for a quantity, a dictionary or "code space" for a category, a spatial reference system for location and geometry, and a temporal reference system for time values. An observed value may be semantically typed according to the phenomenon being observed or observable, sometimes called measurand. Observed values may have other properties, such as quality indicators.

11.3.3 Implementations

There are five known implementations of SensorML actively involved in the community.

11.3.4 Recommendation

Level 2-3: In the prototyping stage.

Although a tremendous amount of interest has occurred, SensorML is still in the middle stages of the specification process.

12 Preliminary Conclusions and Recommendations

This report represents the analysis of current Web Services specifications, standards, and proposed standards emerging primarily from commercial industry consortiums, with a focus on standards that are relevant to the development of next generation DoD Command and Control (C2) systems. This section is called "preliminary" because it summarizes the initial conclusions and recommendations drawn from the initial analysis effort. These conclusions will be augmented by the results of follow-up analysis tasks exploring C2 user requirements and the emerging Web Services development environment.

In general, we believe Web Services are usable and useful today, but implementers must get past the general myth that the current Web Services standards guarantee interoperability. Interoperability is enhanced by these standards, mainly through simplification, such as using ubiquitous communications channels (HTTP), and a simplified, man-readable, yet structured and flexible data format (XML). However, the key to interoperability is the semantics of the connection. Additional standards including XML Schema, SOAP, and WSDL also add value, but still do not clearly convey the semantics without human intervention. To solve the problem additional specifications (including application-specific specifications) need to be adopted, but thanks to the benefits of Web Services, the scope of these specifications can be narrowed.

We believe the DoD should continue and expand upon current efforts to use Web Services standards as an alternative to the stove-piped, proprietary, and often platform-centric means of creating systems. Providing a more open environment to support access to the services and data of C2 systems will foster new and more creative solutions leveraging a wide array of "knowledge" available from sensors and databases.

The following statements summarize, based on the topic areas of this report, other conclusions and recommendations with regard to the Web Services standards environment:

- "Base" Web Services Standards

  In terms of the "base" Web Services standards (such as Web Services Definition Language—WSDL—and Simple Object Access Protocol—SOAP), we are seeing these standards advance within W3C. These standards are being adopted broadly but still show some immaturity in that they don't yet guarantee unambiguous interoperability. There is ongoing work in this area to refine the specifications, from within and without.

- Web Services Architectures/Frameworks

  As adoption of Web Services has grown, the need to define more concrete architectures has grown as well. There are multiple efforts to do so that bear close watching. We believe that the major effort in this area is the W3C Web Services Architecture.

  The Global XML Web Services Architecture (GXA) specifications looked promising at one point, but we are somewhat concerned about the slow pace at which the GXA specifications are being advanced into open standards consortiums. However, we believe that they bear close watching because of the wide range of functionality that they cover, and because one of the GXA specifications (WS-

Security) has already been transferred into an open standards consortium (OASIS).

We do not recommend that DISA adopt the ebXML framework as a whole, but instead consider individual specifications such as ebXML Registry.

- Web Services Security

  A large number of open standards (and potential open standards) are currently emerging in the realm of security. We foresee the current lack of overall robust security for Web Services improving greatly in the next two years, as many of the specifications mature and others arise. The current lack of overall robust security makes it difficult to execute Web Services scenarios that stretch beyond "point-to-point" interactions; we therefore recommend that DISA utilize Web Services at this time, but in point-to-point interactions using established mechanisms such as traditional Public Key Infrastructure (PKI) and Secure Socket Layer/Transport Layer Security (SSL/TLS).

  We believe that the emerging OASIS WS-Security specification will have the largest single impact on the advancement of Web Services of any of the current Web Services security specifications.

  We recommend that OASIS Security Assertion Markup Language (SAML) be used at this time, as it has gained wide acceptance in many different industries and settings. We foresee the number of SAML implementations growing steadily in the medium- and long-term future.

- Web Services Choreography and Coordination

  The area of choreography and coordination is a relatively new and much-anticipated area for Web Services. We believe that advancements in this area will advance Web Services to new heights that will enable inter-organization and inter-agency collaborations.

  We believe that the work of the W3C Web Services Choreography Working Group is highly important and bears close watching, as is the emerging work of the OASIS Web Services Business Process Execution Language (WS BPEL) Technical Committee.

- Web Services and Discovery

  We believe that Universal Description, Discovery and Integration (UDDI) serves a very important purpose, for both DISA and the federal government, as usage of Web Services increases, and its adoption is currently on the rise. However, the current UDDI specifications (versions 2.0 and 3.0) are still somewhat immature and are not specific enough to guarantee portability or interoperability. The weaknesses can be overcome by limiting the use of UDDI implementations to only standard features and augmenting the UDDI usage by adopting standard practices to achieve the goals for discovery.

  Also, the taxonomies for UDDI discovery are (intentionally) not standardized and typically very limited in their ability to represent complex queries. In the future,

we feel that this is a natural fit with some of the emerging ontology standards (e.g., OWL), which could be used to allow searches based on concepts rather than on specific terms that must now be matched exactly.

Because the UDDI Version 3.0 specification is still under development, we recommend that DISA consider using UDDI Version 2.0 specification implementations for the time being in order to advance its Web Services efforts, and upgrade to Version 3.0 when it becomes an OASIS standard and when an acceptable number of implementations are available.

We believe that the current adoption of ebXML Registry has, in general, been very low. However, as adoption of XML grows both within DISA and the federal government—particularly the creation of XML schemas—we foresee the need for an XML registry such as ebXML Registry increasing. Once the OASIS/ebXML Registry Version 2.5 specifications reach a Version 3.0 status, an assessment should be made regarding available implementations and further consideration should be given to implementing ebXML Registry.

We believe that ebXML Registry and UDDI will co-exist and continue to be utilized in their areas of strength: ebXML Registry for discovery and collaboration, and UDDI for discovery.

- Web Services and Reliable Messaging

  We believe that reliable messaging is a necessity for Web Services. At this time, the only open standard that addresses reliable messaging is the ebXML Messaging Service (ebMS) specification, but there has not been widespread adoption of this standard yet. At the moment, vendor-specific MOM products dominate the solution space. There are several vendor specifications that have recently emerged, and an OASIS Technical Committee (Web Services Reliable Messaging—WSRM) is in the process of creating an open standard for reliable messaging. This area bears close watching, particularly to see which specifications emerges as the leader from the current set of specifications.

- Web Services Interoperability

  The Web Services Interoperability Organization (WS-I) is the leading organization that defines how to achieve interoperability between Web Services standards. We believe its work is highly important, but recommend that DISA hold off from utilizing any of its profiles (such as the WS-Basic Profile) until more vendor implementations emerge.

- Semantic Web Services

  The area of Semantic Web Services is producing specifications (such as the OWL Web Service Ontology Language, OWL-S) that are currently in the research stages, but bare close watching.

- Web Services Monitoring and Management

  This is an emerging area that we believe will have highly important results for Web Services. The OASIS Web Services Distributed Management (WSDM) TC

is in the process of creating specifications that will be released in January 2004. We recommend that DISA evaluate these specifications once they are released.

- Standards Related to Applications of Web Services

  In the area of user-facing technologies, we believe that OASIS Web Services for Remote Portlets (WSRP) will have a large impact on Web Services through its ability to seamlessly deliver aggregated content to a centralized location. There has also been a strong emphasis in the federal government on portal technology in recent years, which makes WSRP even more attractive. However, although a fair number of WSRP implementations are available, we believe that the freshness of this standard warrants a waiting period before use.

  In Command and Control, the "map"—i.e., geospatial data and displays—are critical and pervasive in C2 systems. The OGC, an international consortium with significant DoD involvement, has developed a number of standards for geospatial data sharing and processing, primarily using Web Services technologies. DoD system acquisition continues to migrate more and more toward using COTS products versus building solutions. We believe these OGC standards will play a key role both in promoting interoperability between C2 systems, as well as in supporting competitive procurement by providing the Government options when selecting among competing geospatial products.

Appendix A – Referenced Online Content

The following pages reflect the content of some of the web pages referenced in this document.

URL: http://lists.w3.org/Archives/Public/www-ws-arch/2003Aug/0047.html
Subject: RE: Definition for a Web Service
From: Cutler, Roger (RogerCutler) <RogerCutler@chevrontexaco.com>
Date: Mon, 11 Aug 2003 14:42:54 -0500
Message-ID:
<7FCB5A9F010AAE419A79A54B44F3718E01817F45@bocnte2k3.boc.chevrontexaco.net>
To: "Anne Thomas Manes" <anne@manes.net>, "Jean-Jacques Moreau" <jean-jacques.moreau@crf.canon.fr>, www-ws-arch@w3.org

We have -- and I personally think that this is unfortunate but it does represent a clear, consensus-driven decision by the WG that I accept, albeit reluctantly -- limited the scope of what we are willing to call Web Services and discuss in our architecture to thingies that are described by WSDL and use SOAP -- as you can see in the definition. That is, as far as we are concerned thingies described (only) by text documents or DAML (unless DAML is somehow integrated into WSDL, which I understand may not be an unreasonable expectation) are not Web Services. This was a highly contentious issue and the resolution of it was so difficult that I think it would take some sort of dramatic change in the situation to convince people in the WG to reopen it. As I said, I don't like this resolution, but I would like reopening the issue a WHOLE LOT LESS!

That was not, however, the thrust of your message. I personally agree that Web Services are "important" resources and, for that reason, should be identified by a URI. I do not know how many others on the WG would also agree, but I would guess at least some. Or at least would agree that "it sure would be nice" if Web Services were identified by a URI.

It is my perception that the WG is, in effect, unwilling to do things that are not compatible with what the WS-Desc WG is doing/has done, and is also unwilling to tell the WS-Desc WG what to do. I would be very surprised, however, if anyone on the WSA-WG would actually object violently if the WS-Desc WG were somehow to decide to use URI's to identify Web Services.

Obviously the comments above are my personal take on the situation ... Another member of the WG might view things quite differently and I am in no way a spokesman for the WG.

-----Original Message-----
From: Anne Thomas Manes [mailto:anne@manes.net]
Sent: Monday, August 11, 2003 1:02 PM
To: Cutler, Roger (RogerCutler); Jean-Jacques Moreau; www-ws-arch@w3.org
Subject: Re: Definition for a Web Service

I raised a discussion on the WS-Desc list suggesting that they really should identify a Web Service by a URI rather than just a Qname. I was a little surprised by the resistence to such a concept. I got the sense that a lot of people didn't understand what in fact the URI was meant to identify.

I don't know what the end decision on the discussion was. I believe it was discussed at the last meeting.

But I do think that the architecture group should have some influence on the discussion. If the architecture group believes that a Web Service should be named by a URI, then the WS-Desc team should provide a means to capture that name in the WSDL description.

From my perspective, a Web Service is an "important" resource, and as the Web Architecture says, all "important" resources should have a URI.

I also expect that a Web Service may be described by a variety of description languages (WSDL, DAML, text documents, etc.) and so there ought to be a means of referring to the Web Service that doesn't depend on just one description language (a URI derived from the wsdl:service Qname).

Anne

----- Original Message -----
From: "Cutler, Roger (RogerCutler)" <RogerCutler@chevrontexaco.com>
To: "Jean-Jacques Moreau" <jean-jacques.moreau@crf.canon.fr>;
<www-ws-arch@w3.org>
Sent: Monday, August 11, 2003 10:47 AM
Subject: RE: Definition for a Web Service

>
> I think that this happened because of all the confusion about URI's
> and QNames. As I understand it (and I am very willing to admit that I

> understand this imperfectly), just about everyone concerned would be
> VERY happy to say that Web Services are identified by URI's -- except
> that currently in WSDL they are identified by a Qname -- which is not
> exactly a URI but can be mapped to a URI. This, at the least, adds a
> layer of confusion to any conversation on this subject. I think that
> the basic thinking was that the "Web-related standards" would lead one

> sort of inevitably to URI's, and that the detailed issues could be
> dealt with ... in the detailed sections, I guess.
>
> -----Original Message-----
> From: Jean-Jacques Moreau [mailto:jean-jacques.moreau@crf.canon.fr]
> Sent: Monday, August 11, 2003 3:45 AM
> To: www-ws-arch@w3.org
> Subject: Definition for a Web Service
>
>
>
> Thanks for the new draft; obviously, this is the result of a lot of
> efforts!
>
> Regarding the new definition for a Web Service: apart from being more
> specific (WSDL, SOAP, HTTP), which I like, the other major difference
> seems to be that a Web Service is no longer identified by a URI. Is
> this
>
> intentional? Shouldn't this be added back?
>
> <previousDefinition>
> A Web Service is a software system identified by a URI [...].
> </previousDefinition>
>
> Comments?
>
> Jean-Jacques.
>
> Champion, Mike wrote:
>
> > Update from the W3C publication team:
> >
> > New WD of "Web Services Architecture" Document is available at :
> > http://www.w3.org/TR/2003/WD-ws-arch-20030808/

URL: http://lists.w3.org/Archives/Public/www-ws-arch/2003Sep/0086.html
Subject: Myth of Loose coupling
From: David Orchard <dorchard@bea.com>
Date: Fri, 26 Sep 2003 18:01:02 -0700
To: <www-ws-arch@w3.org>
Message-ID: <012501c3851f$be6cfd40$470ba8c0@beasys.com>

I'm posting a link as I was asked to before on the start of a discussion on loose coupling.

http://lists.w3.org/Archives/Public/www-ws-arch/2003Jan/0115.html

I will say that I have come to have a somewhat revised view on loose coupling. I would say that loose coupling is a combination of properties:
- extensibility, so that additional information can be added without breaking receivers
- evolvable changes in the interface, so compatible changes can be made.
- rapidity of changes in the interface
- on the web, the generic interface constraint, means that applications (browsers/search engines) are not dependent upon each site's protocol.
- asynchrony, so that senders and receivers are decoupled in time
- stateless messaging, so that senders need fewer messages and hence less chance of communication errors
- use of URIs for identifying resources. This means that identifiers are very constrained and easily transferred.
- No vendor specific or platform specific constraints on any of the technologies used.

I think one can then say that loose coupling is a property that is a combination of other properties as I've listed above. And it seems that changing each property/constraint increases the coupling. For example, a Web Service with no extensibility, that evolves rapidly in incompatible ways, an application specific interface, synchronous, stateful messages is tightly coupled with it's clients.

This would show that the Web is "mostly" loosely coupled because of the extensibility/evolvability in http/html, slow changes in html vocabularies, stateless messaging, vendor/platform agnostic. Yet it is tightly coupled in being synchronous.

Another way of looking at this is that Web Service technologies do not per se mean a service is loosely coupled, it is only through the application of constraints to be loosely coupled.

Seem reasonable?

I think this notion of a "combination" property is similar to the visibility property, which I argue is a combination of simplicity and percieved performance properties.

Cheers,
Dave